A User Guide to the PROCESS Systems Code

P. J. Knight

EURATOM/CCFE Fusion Association
Culham Science Centre, Abingdon, Oxon, OX14 3DB, UK

$Date:: 2013-04-16 15:58:27 +0100 (Tue, 16 Apr 2013) $ $Revision:: 163 $

Contents

1 Introduction **6**

1.1 Rationale . . . 6

1.2 History . . . 7

1.3 Layout of the User Guide . . . 7

2 Program Overview — The Fundamental Concepts **8**

2.1 Equation Solvers . . . 8

2.1.1 Non-optimisation mode . . . 8

2.1.2 Optimisation mode . . . 8

2.1.3 Scans . . . 10

2.2 The Variable Descriptor File . . . 10

2.3 Input Parameters . . . 12

2.4 Constraint Equations . . . 12

2.4.1 Consistency equations . . . 12

2.4.2 Limit equations . . . 12

2.5 Iteration Variables . . . 13

2.6 Figures of Merit . . . 15

2.7 Scanning Variables . . . 15

2.8 Code Structure . . . 15

2.8.1 Numerics modules . . . 18

2.8.2 Physics modules . . . 18

2.8.3 Engineering modules . . . 18

2.8.4 Costing module . . . 18

2.8.5 Other modules . . . 21

3 Physics, Engineering and Other Models **22**

3.1 Tokamak Power Plant . . . 22

3.1.1 Radial and vertical build . . . 22

3.1.2 Plasma physics models . . . 25
3.1.3 First wall . . . 30
3.1.4 Divertor . . . 30
3.1.5 Blanket . . . 30
3.1.6 Shield . . . 31
3.1.7 Toroidal field coils . . . 32
3.1.8 Poloidal field coils . . . 34
3.1.9 Ohmic heating coil . . . 36
3.1.10 Auxiliary power systems: heating and current drive . . . 36
3.1.11 Structural components . . . 37
3.1.12 Power conversion and heat dissipation systems . . . 38
3.1.13 Cryostat and vacuum system . . . 39
3.1.14 Buildings . . . 39
3.2 Tight Aspect Ratio Tokamak Model . . . 39
3.2.1 Tight aspect ratio tokamak switches . . . 40
3.3 Pulsed Plant Operation . . . 40
3.3.1 Thermal cycling package . . . 40
3.3.2 First wall coolant temperature rise limit . . . 41
3.3.3 First wall peak temperature limit . . . 41
3.3.4 Start-up power requirements . . . 41
3.3.5 OH coil swing time . . . 41
3.3.6 Burn time . . . 41
3.3.7 Thermal storage . . . 41
3.4 Hydrogen Production Facility . . . 42
3.5 Stellarator Model . . . 42
3.5.1 Stellarator physics . . . 43
3.5.2 Machine configuration . . . 44
3.5.3 Limitations in the model . . . 45
3.6 Reversed Field Pinch Model . . . 48
3.6.1 RFP physics . . . 48
3.6.2 TF coils . . . 49
3.6.3 OH coils . . . 49
3.6.4 EF coils . . . 49
3.6.5 Divertor . . . 49
3.6.6 Code modifications . . . 50

3.7 Inertial Fusion Energy Model . . . 50
3.8 Safety and Environment Models . . . 51
3.8.1 Neutronics . . . 51
3.8.2 Activation and inventory information . . . 51
3.9 Cost Models . . . 51
3.9.1 Cost options . . . 52
3.10 Other Switches and Models . . . 53
3.10.1 Output control . . . 53
3.10.2 Code parameters affecting other models . . . 53

4 Execution of the Code 54
4.1 The Input File . . . 54
4.1.1 Tokamak, stellarator, RFP or IFE? . . . 54
4.1.2 File structure . . . 54
4.1.3 Format rules . . . 55
4.2 The Output File . . . 56
4.3 Running the Code . . . 56
4.3.1 Non-optimisation mode . . . 56
4.3.2 Optimisation mode . . . 58
4.4 Problem Solving . . . 59
4.4.1 General problems . . . 59
4.4.2 Optimisation problems . . . 60
4.4.3 Unfeasible results . . . 60
4.4.4 Hints . . . 62

5 Inclusion of Additional Variables and Equations 63
5.1 Input Parameters . . . 63
5.2 Iteration Variables . . . 63
5.3 Other Global Variables . . . 64
5.4 Constraint Equations . . . 64
5.5 Figures of Merit . . . 65
5.6 Scanning Variables . . . 65

6 Code Management Utilities 66
6.1 Makefile . . . 66
6.2 Automatic Documentation . . . 66

7 Acknowledgements & Bibliography **67**

A Non-optimisation Input File **71**

B Optimisation Input File **74**

C Source Code Documentation **77**

Chapter 1

Introduction

1.1 Rationale

During the course of studies into a proposed fusion power plant, there may be times when questions of the following type arise:

> Are the machine's physics and engineering parameters consistent with one another?
>
> Which machine of a given size and shape produces the cheapest electricity?
>
> What is the effect of a more optimistic limit on the maximum plasma density on the amount of auxiliary power required?

Questions such as these are extremely difficult to answer, since the large number of parameters involved are highly dependent on one another. Fortunately, computer programs have been written to address these issues, and `PROCESS` is one of them.

Suppose that an outline power plant design calls for a machine with a given size and shape, which will produce a certain net electric power. There may be a vast number of different conceptual machines that satisfy the problem as stated so far, and `PROCESS` can be used in "non-optimisation" mode to find one of these whose physics and engineering parameters are self-consistent. However, the machine found by `PROCESS` in this manner may not be possible to build in practice — the coils may be overstressed, for instance, or the plasma pressure may exceed the maximum possible value. `PROCESS` contains a large number of constraints to prevent the code from finding a machine with such problems, and running the code in so-called "optimisation" mode forces these constraints to be met. The number of possible conceptual machines is thus considerably reduced, and optimisation of the parameters with respect to (say) the cost of electricity will reduce this number to a minimum (possibly one).

Formally then, `PROCESS` is a systems code that calculates in a self-consistent manner the parameters of a fusion power plant with a specified performance, ensuring that its operating limits are not violated, and with the option to optimise a given function of these parameters.

It would not be fair to call `PROCESS` a fusion power plant design code, as this implies that a great deal of complexity would need to be present in each and every model describing one of the component systems. Such complexity is, however, incompatible with the code's iterative approach to solving the optimisation problem, since this requires repeated evaluation of the same (large number of) expressions. This is not to say that the models employed by the code are oversimplified — in general they represent good numerical estimates of present theoretical understanding, or are fits to experimental data. `PROCESS` provides a useful overall description of how a conceptual and feasible power plant may look.

1.2 History

PROCESS is derived from several earlier systems codes, but is largely based on the TETRA (Tokamak Engineering Test Reactor Analysis) code [1] and its descendant STORAC (Spherical TOrus Reactor Analysis Code) [2], which includes routines relevant to the tight aspect ratio class of tokamaks. These codes, and much of the original version of PROCESS itself, were written by personnel at Oak Ridge National Laboratory in Tennessee, USA, with contributions from a number of other laboratories in the USA. In addition, many of the mathematical routines have been taken from a number of different well-established source libraries.

Since the code is descended from such a wide range of sources, its structure was initially not ideal from the programmer's viewpoint. Non-standard practices and inconsistent layout within the code led to difficulties in modifying, interpreting and indeed running the code. A great deal of effort was therefore expended at Culham on the code's arrival from ORNL in the early 1990s to improve this situation, with the code being given a complete but careful upgrade, routine by routine. For many years this Fortran 77 code was used for systems code studies of various power plant scenarios, and was modified from time-to-time by the addition of new and/or improved models, including machines based on the stellerator, reversed field pinch and inertial confinement concepts.

In 2012, the code structure was revised again to allow it to benefit from modern software practices, and the whole program was upgraded to Fortran 90/95. At the same time a number of useful code management utilities were added.

As with all active research codes, PROCESS will continue to be developed into the future. This User Guide is updated in parallel with the Fortran source code itself to ensure that the documentation remains consistent with the latest version of the code. It is to be hoped that it will be of assistance to all users of PROCESS, whether they are planning to modify or run the code, or are simply trying to understand what the code aims to achieve.

1.3 Layout of the User Guide

The User Guide is divided into a small number of logically separate chapters, each one of which provides specific information on a given topic. It depends on the user's motive for referring to the manual as to which chapter will be the most useful, although hopefully the style and structure adopted will allow one to browse through without difficulty.

Chapter 2 provides an overview of the program, and outlines the numerical and programming concepts involved. Chapter 3 describes the physics, engineering and economic models that are used within the code, and lists the switches available allowing the user to customise the models' details to achieve the desired simulation. Chapter 4 describes how to run the program from scratch, and provides a number of hints and suggestions for the user to bear in mind to help the code find a feasible machine. Chapter 5 shows how to modify the code in specific ways, for example how to add extra constraints and variables to the code. A useful set of code management utilities is introduced in Chapter 6. Finally, the Appendices give example input files for PROCESS in non-optimisation and optimisation modes, and lists of references that provide information about the code status, its location, and other details relating to the implementation of PROCESS to date.

Chapter 2

Program Overview — The Fundamental Concepts

Fusion power plants are complex systems consisting of many non-linear interactions. One method that can be used to model this kind of system is to iterate a number of free parameters (the so-called *iteration variables* — see Section 2.5) in a controlled way so as to find a self-consistent set of device parameters that satisfy all of the system's *constraint equations* — see Section 2.4. PROCESS is organised in a standard equation solver format to enable this task to be performed efficiently. The physics and engineering routines together serve as a *function evaluator*, providing the information used in the solution of the constraints. The numerical modules present in PROCESS perform the iteration required, and also incorporate the option to maximise or minimise a given *figure of merit* – see Section 2.6.

2.1 Equation Solvers

PROCESS contains two non-linear equation solver packages, which reflect the two major modes of operation available. Each of these has its own uses, as is now discussed.

2.1.1 Non-optimisation mode

The first of the two equation solvers present in PROCESS is the non-optimisation package HYBRID [11, 12]. Formally, HYBRID finds a zero of a system of N non-linear functions in N variables. This means simply that N variables (power plant parameters) are iterated by PROCESS in such a way as to solve a set of N equations (physics or engineering laws), i.e. a set of self-consistent power plant parameters is found. This is useful for performing benchmark comparisons, when the device size is kept fixed, and one only wishes to find calculated stresses, beta values, fusion powers, etc. A flow diagram of PROCESS in non-optimisation mode is shown in Figure 2.1.

2.1.2 Optimisation mode

The HYBRID equation solver will naturally find only one of perhaps several possible machines that may satisfy the prescribed problem. To choose one machine in preference to the others it is necessary to define a *figure of merit*, and the selection process then simply involves finding the machine parameters that maximise or minimise this figure of merit. The second equation solver within PROCESS, VMCON [13], performs this optimisation, and therefore finds the "best" machine that satisfies all the

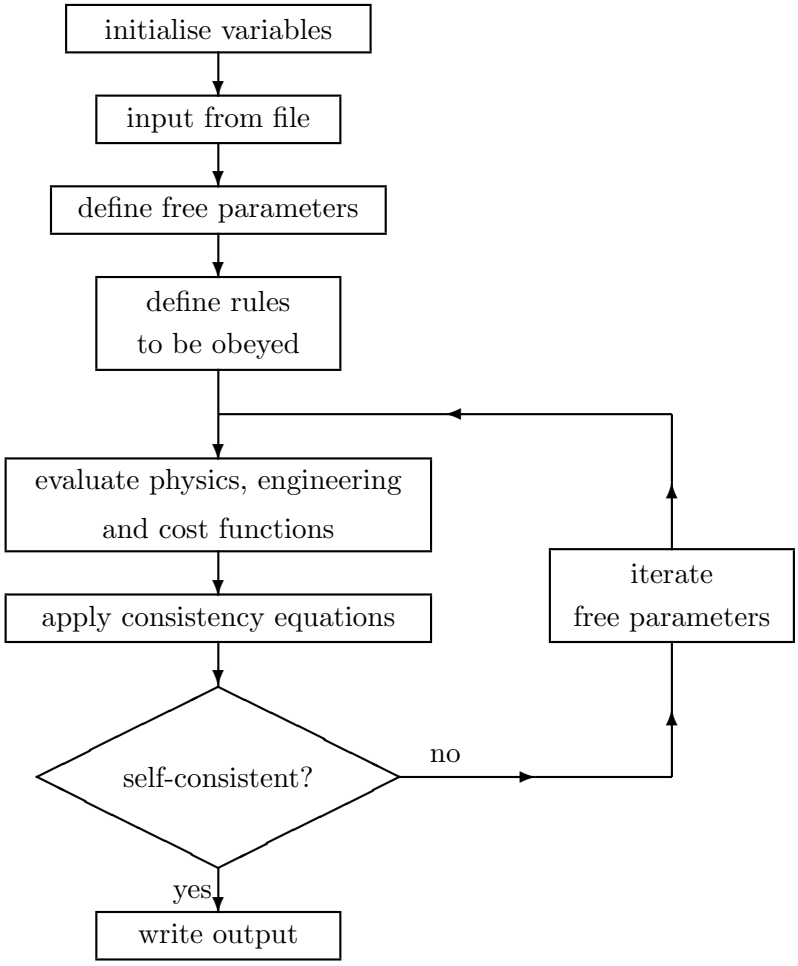

Figure 2.1: *Flow diagram of* `PROCESS` *in non-optimisation mode.*

given constraints.

An important advantage that VMCON has over HYBRID is its ability to limit the ranges of the variables it uses. This prevents the code from attempting to find machines that are physically unattainable, and ensures that operating limits are not violated. An example of VMCON's application is to find the device providing the minimum cost of electricity which also satisfies the physics and engineering constraints. There is in theory no upper limit to the number of variables that VMCON can use to optimise the machine, so a very large region of parameter space can be searched. A flow diagram of `PROCESS` in optimisation mode is shown in Figure 2.2.

2.1.3 Scans

It is often useful to be able to scan through a range of values of a given parameter to see what effect this has on the machine as a whole. Sensitivity studies of this kind can be achieved very easily using `PROCESS`. Scans are carried out in optimisation mode, whereby the code performs initially a run using the parameters specified in the input file, and then a series of runs using the parameters produced at the end of the previous iteration. The value of the quantity being scanned is specified at every stage — see Section 2.7. This method ensures that a smooth variation in the machine parameters is achieved.

2.2 The Variable Descriptor File

The variable descriptor file `vardes.html` is an invaluable resource for the user of `PROCESS`. It acts as a dictionary / reference manual for the code's variables, and contains the following information about each:

- name
- dimensions (of arrays)
- default value(s) of those variables that are not initially derived from a combination of other values. The default values are mostly set in the modules contained within source file `global_variables.f90`.
- description, including physical units if relevant
- for switches/flags, the meanings of all allowed values
- iteration variable number, if relevant
- corresponding constraint equation, if relevant

In addition, global code parameters are labelled `FIX`. These can only be changed by editing the relevant source file, but this should not be carried out unless it is absolutely necessary.

All the variables that are shown with a default value are available to be changed by the user using the input file (Section 4.1), except for those which are labelled `FIX`. Variables not shown with a default value are calculated by the code from a combination of other parameters, and so it would be meaningless to initialise them. Obviously, these variables cannot be changed using the input file.

The file is generated from specially-formatted comment lines within the source code (see Section 6.2 for more details). Therefore, it is exceedingly important to keep these comment lines relevant and in sync with the variables they describe.

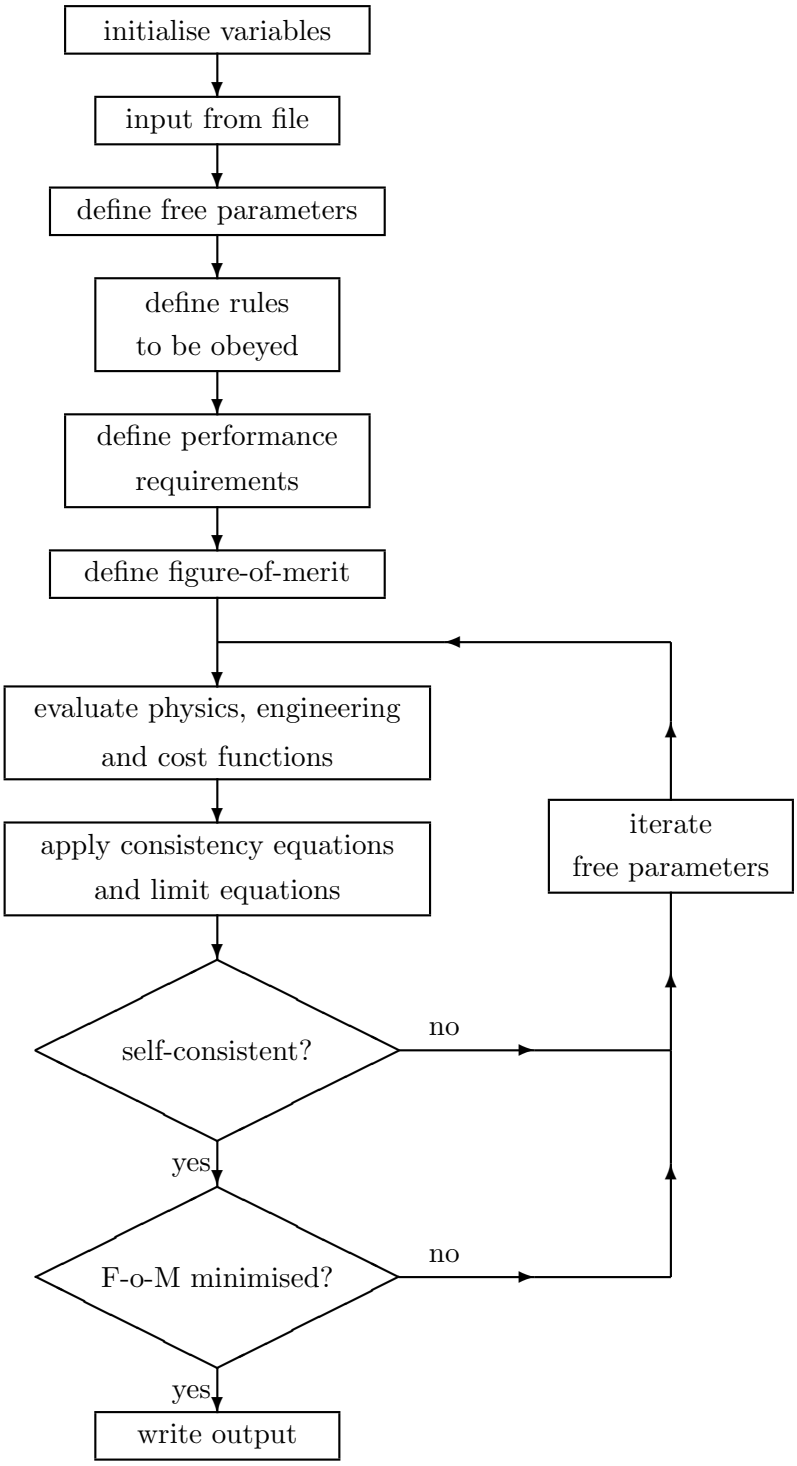

Figure 2.2: *Flow diagram of* `PROCESS` *in optimisation mode.*

2.3 Input Parameters

Input parameters make up a large proportion of the variables listed in the variable descriptor file. They comprise all those variables that, once set in the initialisation routine or redefined in the input file, do not change throughout a PROCESS run. In fact, only those variables defined as iteration variables (Section 2.5) can change during the course of a run.

2.4 Constraint Equations

Any computer program naturally contains myriads of equations. The built-in equation solvers within PROCESS act on a special class, known as *constraint equations*, all of which are formulated in routine CONSTRAINTS in source file constraint_equations.f90. Table 2.1 summarises the constraint equations available in PROCESS. These can be split into two types — (1) consistency equations, that enforce consistency between the physics and engineering parameters, and (2) limit equations, that enforce various parameters to lie within their allowed limits. The neqns constraint equations that the user chooses for a given run are activated by including the equation numbers in the first neqns elements of array icc.

2.4.1 Consistency equations

Consistency equations are usually *equalities* that ensure that the machine produced by PROCESS is self-consistent. This means, therefore, that many of these constraint equations should *always* be used, namely equations 1, 2, 10 and 11 (see Table 2.1). Equation 7 should also be activated if neutral beam injection is used and equation 15 should be used if a pulsed machine (lpulse = 1) is being modelled. The other consistency equations can be activated if required.

A typical consistency equation ensures that two functions g and h are equal:

$$\begin{aligned} g(x, y, z, \ldots) &= h(x, y, z, \ldots) \\ c_i &= 1 - \frac{g}{h} \end{aligned}$$

The equation solvers VMCON and HYBRID need the constraint equations c_i to be given in the form shown, since they adjust the iteration variables so as to obtain $c_i = 0$, thereby ensuring that $g = h$.

2.4.2 Limit equations

The limit equations are usually *inequalities* that ensure that various physics or engineering limits are not exceeded. Each of these equations has an associated *f-value*, which allow them to be coded as equalities. The f-values are used as follows.

In optimisation mode, all iteration variables have prescribed lower and upper bounds. In general, limit equations have the form

$$\textit{calculated quantity} = f \times \textit{maximum allowable value}$$

where f is the f-value. If f has a lower bound of zero and an upper bound of one, then the limit equation does indeed constrain the calculated quantity to lie between zero and its maximum allowable value, as required.

As with the consistency equations, the general form of the limit equations is

$$c_i = 1 - f.\frac{h_{\max}}{h}$$

where $h_{\max}$ is the maximum allowed value of the quantity h.

Sometimes, the limit equation and f-value are used to ensure that quantity h is *larger* than its *minimum* value $h_{\min}$. In this case, $0 \leq f \leq 1$ (as before), but the equation takes the form

$$c_i = 1 - f.\frac{h}{h_{\min}}$$

By fixing the f-value (i.e. not including it in the `ixc` array), the limit equations can be used as equality constraints. For example, to set the net electric power to a certain value, the following should be carried out:

1. Activate constraint equation 16 by including it in the first `neqns` elements of array `icc`
2. Set `fpnetel = 1.0D0`
3. Ensure that `fpnetel` (iteration variable no. 25) *DOES NOT* appear in array `ixc`
4. Set `pnetelin` to the required net electric power.

Limit equations are not restricted to optimisation mode. In non-optimisation mode, the iteration variables are not bounded, but the f-values can still be used to provide information about how calculated values compare with limiting values, without having to change the characteristics of the device being benchmarked to find a solution.

It is for this reason that all the constraint equations used in `PROCESS` are formulated as equalities, despite the fact that equation solver VMCON can solve for inequalities as well. The use of f-values precludes this need, and allows the non-optimising equation solver HYBRID to use the same constraint equations.

2.5 Iteration Variables

It is necessary to calculate numerical derivatives during the solution of the constraint equations. The iteration variables are the parameters that the equation solvers use for this purpose — all the other code variables (input parameters — see above) remain fixed at their initial value. Successive calls are made to the physics and engineering routines, with slightly different values for the iteration variables on each call, and the equation solver determines the effect on the output due to these small changes to the input (see Figures 2.1 and 2.2). The `nvar` iteration variables that the user chooses for a given run are activated by including the variable numbers in the first `nvar` elements of array `ixc`. Tables 2.2 and 2.3 list the iteration variables available in `PROCESS`.

Clearly, the equation solvers need at least as many variables to iterate as there are equations to solve, i.e. `nvar` $\geq$ `neqns`. If the run is a non-optimising case, then `neqns` variables are iterated — the values of the remaining `(nvar-neqns)` variables are left alone. If the run is an optimising case, then all the active iteration variables are adjusted so as to find the minimum (or maximum) value of a parameter (the *figure of merit*) in the `nvar`-dimensional space of the problem.

All the iteration variables are constrained to lie between lower and upper bounds, stored in arrays `boundl` and `boundu`, respectively. For instance, the plasma electron density is, by default, confined to

`icc` no.	description	type	corresponding `ixc` variables
1	plasma beta consistency	C	**5**
2	global power balance	C	**10**,1,2,3,4,6,11
3	ion power balance	C	**10**,1,2,3,4,6,11
4	electron power balance	C	**10**,1,2,3,4,6,11
5	density limit	L	**9**,1,2,3,4,5,6
6	epsilon-beta poloidal limit	L	**8**,1,2,3,4,6
7	beam ion density (NBI)	C	**7**
8	neutron wall load limit	L	**14**,1,2,3,4,6
9	fusion power limit	L	**26**,1,2,3,4,6
10	toroidal field 1/R consistency	C	**12**,1,2,3,13
11	radial build consistency	C	**3**,1,13,16,29,42,61
12	volt second limit	L	**15**,1,2,3
13	burn time limit (PULSE)	L	**21**,1,16,17,22,29,42,44,61
14	beam energy (NBI)	C	**19**,1,2,3,6
15	burn time (PULSE) (N.B. not usually necessary)	C	**41**,44
16	net electric power limit	L	**25**,1,2,3
17	stellarator radial build (STELLARATOR)	C	**61**
18	divertor heat load limit	L	**27**
19	MVA limit	L	**30**
20	port size limit	L	**33**,31,3,13
21	minor radius limit	L	**32**
22	divertor collisionality limit	L	**34**,43
23	TF coil current density limit	L	**28**,12,24,20,23
24	Troyon beta limit (also beta limit in stellarators)	L	**36**,1,2,3,4,6,18
25	peak toroidal field limit	L	**35**,3,13,29
26	OH coil current density at End of Flat-top limit	L	**38**,37,41,12
27	OH coil current density at Beginning of Pulse limit	L	**39**,37,41,12
28	energy multiplication Q limit	L	**45**,47,40
29	inboard radial build = specified value	C	**3**,1,13,16,29,42,61
30	injection power limit	L	**46**,47,11
31	TF coil case stress limit (SCTF)	L	**48**,56,57,58,59,60,24
32	TF coil conduit stress limit (SCTF)	L	**49**,56,57,58,59,60,24
33	TF coil $I_{\text{operational}}/I_{\text{critical}}$ limit (SCTF)	L	**50**,56,57,58,59,60,24
34	TF coil dump voltage limit (SCTF)	L	**51**,52,56,57,58,59,60,24
35	TF coil $J_{\text{winding pack}}/J_{\text{protection}}$ limit (SCTF)	L	**53**,56,57,58,59,60,24
36	TF coil temperature margin limit (SCTF)	L	**54**,55,56,57,58,59,60,24
37	current drive gamma limit	L	**40**,47
38	first wall coolant temperature rise limit (PULSE)	L	**62**
39	first wall peak temperature limit (PULSE)	L	**63**
40	minimum injection power for ignition limit (PULSE)	L	**64**
41	OH coil swing time limit (PULSE)	L	**66**,65
42	cycle time limit (PULSE)	L	**67**,65,17
43	average centrepost temperature (TART)	C	**69**,70,13
44	peak centrepost temperature limit (TART)	L	**68**,69,70
45	edge safety factor limit (TART)	L	**71**,1,2,3
46	Ip/Irod limit (TART)	L	**72**,2,60
47	TF coil toroidal thickness (RFP, STELLARATOR)	L	**76**,77,13,3
48	poloidal beta limit	L	**79**,2,3,18
49	reversal parameter < 0 (RFP)	L	**80**,78,3,1
50	IFE repetition rate limit (IFE)	L	**86**
51	Startup volt-seconds consistency (PULSE)	C	**16**,29,3,1

Table 2.1: *Summary of the constraint equations present in* `PROCESS`*. Consistency equations are marked C, limit equations are marked L. Some (non-exhaustive) iteration variable numbers (see Tables 2.2 and 2.3) that directly affect the associated constraint equations are given, the one listed first being the most relevant.*

lie between the values 10^{19} m^{-3} and 10^{21} m^{-3}. Of course, it can also be constrained to lie below the *calculated* density limit, if constraint equation 5 is activated and the f-value `fdene` (iteration variable no. 9) is bounded by the values 0 and 1.

It is important to remember that iteration variables *must never be initialised to zero*. The code will not be able to adjust the variable's value if this is done, and it will stop with an error message.

2.6 Figures of Merit

In optimisation mode, `PROCESS` finds the self-consistent set of iteration variable values that maximises or minimises a certain function of them, known as the *figure of merit*. Several possible figures of merit are available, all of which are formulated in routine `FUNFOM` in source file `evaluators.f90`. Switch `minmax` is used to control which figure of merit is to be used, as summarised in Table 2.4. If the figure of merit is to be minimised, `minmax` should be positive, and if a maximised figure of merit is desired, `minmax` should be negative.

2.7 Scanning Variables

One of a number of variables can be scanned during the course of a `PROCESS` run. This option provides a method of determining the sensitivity of the results to different input assumptions. The user specifies which variable is to be scanned (see Table 2.5) and its required value at each point in the scan. The scanned variable to use is defined by the value of `nsweep`, and the chosen variable's values during the scan are set in array `sweep`.

Runs involving scans of this kind can only be performed in optimisation mode. The results from the previous scan point are used as the input to the next scan point. Routine `SCAN` in source file `scan.f90` stores many of the output quantities in a separate output file (`PLOT.DAT`) for use with a plotting program.

Scanning of derived quantities requires use of the appropriate constraint equations. For instance, if the net electric power is scanned, constraint equation 16 should be employed.

For obvious reasons, the active scanning variable must not also be an active iteration variable.

2.8 Code Structure

The structure of the majority of the code reflects to a certain extent the layout of the machine being modelled. As stated above, a large proportion of the code is simply a description of the underlying physics and engineering issues in terms of numerous expressions and relationships. In effect these define the machine so that the numerical solver within the code can then get to work adjusting the parameters in its search for a self-consistent solution.

It is essential for a program of the size and complexity of `PROCESS` to be modular to a high degree, in order to simplify the tasks of understanding and maintaining the code. The use of Fortran 90/95 modules provides a natural and convenient way for this to be done. The following sections describe briefly the modules into which `PROCESS` is divided.

ixc no.	variable name	description	icc eqn	lower bound	upper bound
1	aspect	plasma aspect ratio		1.100D0	10.00D0
2	bt	toroidal field on axis		0.010D0	100.0D0
3	rmajor	plasma major radius		0.100D0	10.00D0
4	te	electron temperature		5.000D0	500.0D0
5	beta	plasma beta		0.001D0	1.000D0
6	dene	electron density		1.00D19	1.00D21
7	rnbeam	hot beam density / electron density		1.00D-6	1.000D0
8	fbeta	f-value for $\epsilon.\beta_p$ limit equation	6	0.001D0	1.000D0
9	fdene	f-value for density limit equation	5	0.001D0	1.000D0
10	hfact	confinement time H-factor		0.100D0	3.000D0
11	pheat	heating power not used for current drive		1.000D6	1.000D9
12	oacdcp	overall current density in TF coil inboard leg		1.000D5	1.500D8
13	tfcth	TF coil inboard leg thickness		0.100D0	5.000D0
14	fwalld	f-value for wall load limit equation	8	0.001D0	1.000D0
15	fvs	f-value for volt second limit equation	12	0.000D0	1.000D0
16	ohcth	OH coil thickness		0.001D0	1.000D2
17	tdwell	dwell time		0.100D0	1.000D8
18	q	edge safety factor		2.000D0	100.0D0
19	enbeam	neutral beam energy		1.000D0	1.000D6
20	tcpav	average (resistive) TF coil temperature		40.00D0	1.000D3
21	ftburn	f-value for burn time limit equation	13	0.001D0	1.000D0
22	tbrnmn	minimum burn time		0.001D0	1.000D6
23	fcoolcp	coolant fraction of resistive TF coil		0.100D0	0.500D0
24	cdtfleg	TF coil leg overall current density		1.000D4	1.000D8
25	fpnetel	f-value for net electric power limit equation	16	1.000D0	1.000D0
26	ffuspow	f-value for fusion power limit equation	9	0.001D0	1.000D0
27	fhldiv	f-value for divertor heat load limit equation	18	0.001D0	1.000D0
28	fjtfc	f-value for TF coil current density limit equation	23	0.100D0	1.000D0
29	bore	machine bore		0.100D0	10.00D0
30	fmva	f-value for MVA limit equation	19	0.010D0	1.000D0
31	gapomin	minimum gap between outboard shield and vacuum vessel		0.001D0	10.00D0
32	frminor	f-value for minor radius limit equation	21	0.001D0	1.000D0
33	fportsz	f-value for port size limit equation	20	0.010D0	1.000D0
34	fdivcol	f-value for divertor collisionality limit equation	22	0.001D0	1.000D0
35	fpeakb	f-value for peak toroidal field limit equation	25	0.001D0	1.000D0
36	fbetatry	f-value for Troyon beta limit equation	24	0.001D0	1.000D0
37	coheof	OH coil current density at end of flat-top		1.000D5	1.000D8
38	fjohc	f-value for OH coil current at EOF limit equation	26	0.010D0	1.000D0
39	fjohc0	f-value for OH coil current at BOP limit equation	27	0.001D0	1.000D0
40	fgamcd	f-value for current drive gamma limit equation	37	0.001D0	1.000D0
41	fcohbop	OH coil current density ratio BOP/EOF		0.001D0	1.000D0
42	gapoh	gap between OH coil and bucking cylinder		0.000D0	10.00D0
43	cfe0	seeded high-Z impurity fraction		1.00D-6	3.00D-3
44	fvsbrnni	fraction of volt seconds from non-inductive means		0.001D0	1.000D0
45	fqval	f-value for energy multiplication limit equation	28	0.010D0	0.330D0
46	fpinj	f-value for injection power limit equation	30	0.001D0	1.000D0
47	feffcd	current drive efficiency multiplier		0.001D0	1.000D0
48	fstrcase	f-value for TF coil case stress limit equation	31	0.001D0	1.000D0
49	fstrcond	f-value for TF coil conduit stress limit equation	32	0.001D0	1.000D0
50	fiooic	f-value for TF coil operational current limit equation	33	0.001D0	0.500D0

Table 2.2: *Iteration variables 1 to 50 present in* PROCESS*. The f-values correspond to the given constraint equations (see Table 2.1). The other iteration variables are shown in Table 2.3.*

ixc no.	variable name	description	icc eqn	lower bound	upper bound
51	fvdump	f-value for TF coil dump voltage limit equation	34	0.001D0	1.000D0
52	vdalw	allowable TF coil dump voltage		0.001D0	1.000D6
53	fjprot	f-value for TF coil current protection limit equation	35	0.001D0	1.000D0
54	ftmargtf	f-value for TF coil temperature margin limit equation	36	0.001D0	1.000D0
55	tmargmin	minimum allowable TF coil temperature margin		0.001D0	100.0D0
56	tdmptf	dump time for TF coil		10.00D0	1.000D6
57	thkcas	TF coil external case thickness		0.050D0	1.000D0
58	thwcndut	TF coil conduit case thickness		0.001D0	1.000D0
59	fcutfsu	copper fraction of cable conductor		0.001D0	1.000D0
60	cpttf	current per turn in the TF coils		0.001D0	4.000D4
61	gapds	gap between vacuum vessel and inboard shield		0.000D0	10.00D0
62	fdtmp	f-value for 1st wall coolant temperature rise limit equation	38	0.001D0	1.000D0
63	ftpeak	f-value for 1st wall peak temperature limit equation	39	0.001D0	1.000D0
64	fauxmn	f-value for minimum auxiliary power limit equation	40	0.001D0	1.000D0
65	tohs	OH coil swing time		0.100D0	1.000D3
66	ftohs	f-value for OH coil swing time limit equation	41	0.001D0	1.000D0
67	ftcycl	f-value for minimum cycle time limit equation	42	0.001D0	1.000D0
68	fptemp	f-value for maximum centrepost temperature limit equation	44	0.001D0	1.000D0
69	rcool	average radius of centrepost coolant channel		0.001D0	0.010D0
70	vcool	maximum centrepost coolant flow speed at midplane		1.000D0	1.000D2
71	fq	f-value for minimum edge safety factor limit equation	45	0.001D0	1.000D0
72	fipir	f-value for maximum I_p/I_{rod} limit equation	46	0.001D0	1.000D0
73	scrapli	inboard scrape-off layer thickness		0.000D0	10.00D0
74	scraplo	outboard scrape-off layer thickness		0.000D0	10.00D0
75	tfootfi	ratio of TF coil outboard/inboard leg thickness		0.200D0	5.000D0
76	frfptf	f-value for TF coil toroidal thickness limit equation	47	0.001D0	1.000D0
77	tftort	TF coil toroidal thickness		0.050D0	2.000D0
78	rfpth	RFP pinch parameter, Θ		0.010D0	1.800D0
79	fbetap	f-value for poloidal beta limit equation	48	0.001D0	1.000D0
80	frfpf	f-value for RFP reversal parameter limit equation	49	0.001D0	1.000D0
81	edrive	IFE driver energy		1.000D5	5.000D7
82	drveff	IFE driver wall plug to target efficiency		0.010D0	1.000D0
83	tgain	IFE target gain		1.000D0	500.0D0
84	chrad	radius of IFE chamber		0.100D0	20.00D0
85	pdrive	IFE driver power reaching target		1.000D6	2.000D8
86	frrmax	f-value for maximum IFE repetition rate equation	50	0.001D0	1.000D0
87	helecmw	electrical power required for hydrogen production		1.000D0	4.000D3
88	hthermmw	thermal power required for hydrogen production		1.000D0	4.000D3

Table 2.3: *Iteration variables 51 to 88 present in* ***PROCESS****. The f-values correspond to the given constraint equations (see Table 2.1). The other iteration variables are shown in Table 2.2.*

`minmax`	description
±1	plasma major radius
±2	ratio of fusion power to input power
±3	neutron wall load
±4	total TF coil + PF coil power
±5	ratio of fusion power to injection power
±6	cost of electricity
±7	direct cost if `ireactor = 0`; constructed cost otherwise
±8	aspect ratio
±9	divertor heat load
±10	toroidal field on axis
±11	injection power
±12	hydrogen plant capital cost
±13	hydrogen production rate
±14	pulse length

Table 2.4: *Summary of the available figures of merit in* ***PROCESS****. If the figure of merit is to be minimised,* ***minmax*** *should be positive, and if a maximised figure of merit is desired,* ***minmax*** *should be negative.*

2.8.1 Numerics modules

These modules contain the equation solvers, their calling routines and other relevant procedures. Various mathematical routines from a number of standard libraries are also incorporated into these files. Table 2.6 summarises the numerics source file contents.

2.8.2 Physics modules

These modules contain the main physics routines that evaluate the plasma and fusion parameters. Also included here are the routines describing the current drive and divertor systems. Table 2.7 summarises the physics source file contents.

2.8.3 Engineering modules

These modules contain the description of the machine geometry and its major systems, including the PF and TF coil sets, the first wall, blanket and shield, and other items such as the buildings, vacuum system, power conversion and the structural components. Table 2.8 summarises the engineering source file contents.

2.8.4 Costing module

The costing module, `costs.f90`, performs all the cost calculations, including values in M$ for each machine system, and the cost of electricity in m$/kWh.

nsweep	scan variable	description
1	aspect	plasma aspect ratio
2	hldivlim	maximum divertor heat load
3	pnetelin	required net electric power
4	hfact	confinement time H-factor
5	oacdcp	overall current density in TF coil inboard leg
6	walalw	allowable wall load
7	beamfus0	beam-background fusion multiplier
8	fqval	f-value for energy multiplication limit eqn
9	te	electron temperature
10	boundu(15)	upper bound on f-value fvs
11	dnbeta	Troyon g coefficient
12	bscfmax	bootstrap current fraction (use negative values)
13	boundu(10)	upper bound on confinement time H-factor hfact
14	fiooic	f-value for TF coil operational current limit eqn
15	fjprot	f-value for TF coil current protection limit eqn
16	rmajor	plasma major radius
17	bmxlim	maximum toroidal field
18	gammax	maximum current drive gamma
19	boundl(16)	lower bound on OH coil thickness ohcth
20	tbrnmn	minimum burn time (pulsed operation machine)
21	sigpfalw	allowable stress in the PF coils
22	cfactr	plant availability factor
23	boundu(72)	upper bound on f-value fipir
24	powfmax	maximum fusion power
25	kappa	plasma elongation
26	triang	plasma triangularity

Table 2.5: *Summary of the scanning variables available in PROCESS.*

source file	description
caller.f90	calls physics and engineering routines
constraint_equations.f90	defines the constraint equations
evaluators.f90	function evaluators for HYBRID and VMCON packages
iteration_variables.f90	adjusts values of iteration variables
maths_library.f90	miscellaneous 'black-box' maths routines, including HYBRID and VMCON
numerics.f90	numerics array definitions, and calling routines for HYBRID and VMCON packages
scan.f90	performs a parameter scan

Table 2.6: *Summary of the numerics modules in PROCESS.*

source file	description
`current_drive.f90`	current drive efficiency calculations
`divertor.f90`	divertor model calculations
`fispact.f90`	nuclide inventory/activation calculations
`ife.f90`	inertial fusion energy physics/engineering
`physics.f90`	tokamak plasma and fusion calculations
`plasma_geometry.f90`	plasma geometry algorithms
`rfp.f90`	reversed field pinch physics/engineering
`startup.f90`	plasma start-up auxiliary power requirements
`stellarator.f90`	stellarator-relevant physics/engineering

Table 2.7: *Summary of the physics modules in* `PROCESS`.

source file	description
`availability.f90`	plant component lifetimes and overall availability
`buildings.f90`	buildings calculations
`fwbs.f90`	first wall, blanket and shield calculations
`machine_build.f90`	machine build calculations
`pfcoil.f90`	PF coil calculations
`plant_power.f90`	heat transport and power balance calculations
`pulse.f90`	pulsed power plant calculations
`safety.f90`	steady-state temperatures after a LOCA event
`sctfcoil.f90`	superconducting TF coil calculations
`structure.f90`	support structure calculations
`tfcoil.f90`	resistive TF coil calculations
`vacuum.f90`	vacuum system calculations

Table 2.8: *Summary of the engineering modules in* `PROCESS`.

2.8.5 Other modules

These modules perform miscellaneous tasks, such as initialisation of variables and file input / output. File `process.f90` contains the main program, and includes the overall controlling loop.

Table 2.9 summarises these modules.

source file	description
`global_variables.f90`	defines and initialises most shared variables
`initial.f90`	checks self-consistency of input variables and switches
`input.f90`	reads in user-defined settings from input file
`output.f90`	utility routines to format output to file
`process.f90`	main program and top-level calling routines

Table 2.9: *Summary of the remaining modules in* `PROCESS`.

Chapter 3

Physics, Engineering and Other Models

There are a great number of individual models within PROCESS, characterising many different aspects of a fusion power plant. Several of these will always be used by the code and so require no input by the user to activate them. However, in many cases there is a choice of model available, and each of these has its own user-controlled switches or flags. This chapter summarises these models, and indicates their location and interaction within the code, together with the relevant switch settings and required parameter values. The concepts used (iteration variables, constraint equations, etc.), and instructions on how to set switches, etc., are explained fully in Chapter 2.

3.1 Tokamak Power Plant

The default (and most detailed) power plant model in PROCESS is based on the tokamak magnetic-confinement fusion concept. This section describes the models relevant to such a plant in some detail, starting with the plasma at the centre of the fusion power core, and moving outwards to the external components, power subsystems and buildings. However, many of these models are also partly or wholly relevant for the other available machine types, especially outside the fusion power core. The specific details for these alternative models are introduced later in the Chapter.

3.1.1 Radial and vertical build

Figure 3.1 shows schematically the layout of a typical tokamak as modelled by PROCESS. This is the so-called 'build' of the machine — the relative locations of the major components. Their positions are referenced to the (R, Z) coordinate system, where R is the radial distance from the vertical centreline (axis) of the torus, and Z is the vertical distance from the equatorial midplane, about which the machine is assumed to be up-down symmetrical (by default; the vertical build is slightly different for single null plasma devices — see Figure 3.2). Components are often referred to as being 'inboard' or 'outboard', which simply means that they lie at a radius R less than or greater than R_0, respectively, where R_0 is the plasma major radius (rmajor). For the sake of clarity the thicknesses are not drawn to scale, and the space labelled as the divertor does not indicate in any way the actual shape of that component.

Most of the thicknesses shown in Figure 3.1 are input parameters, so are not changed during the course of the simulation. The rest are calculated by the code during execution. In addition, some of the component sizes can be used as *iteration variables* (see Section 2.5) to help in the optimisation process.

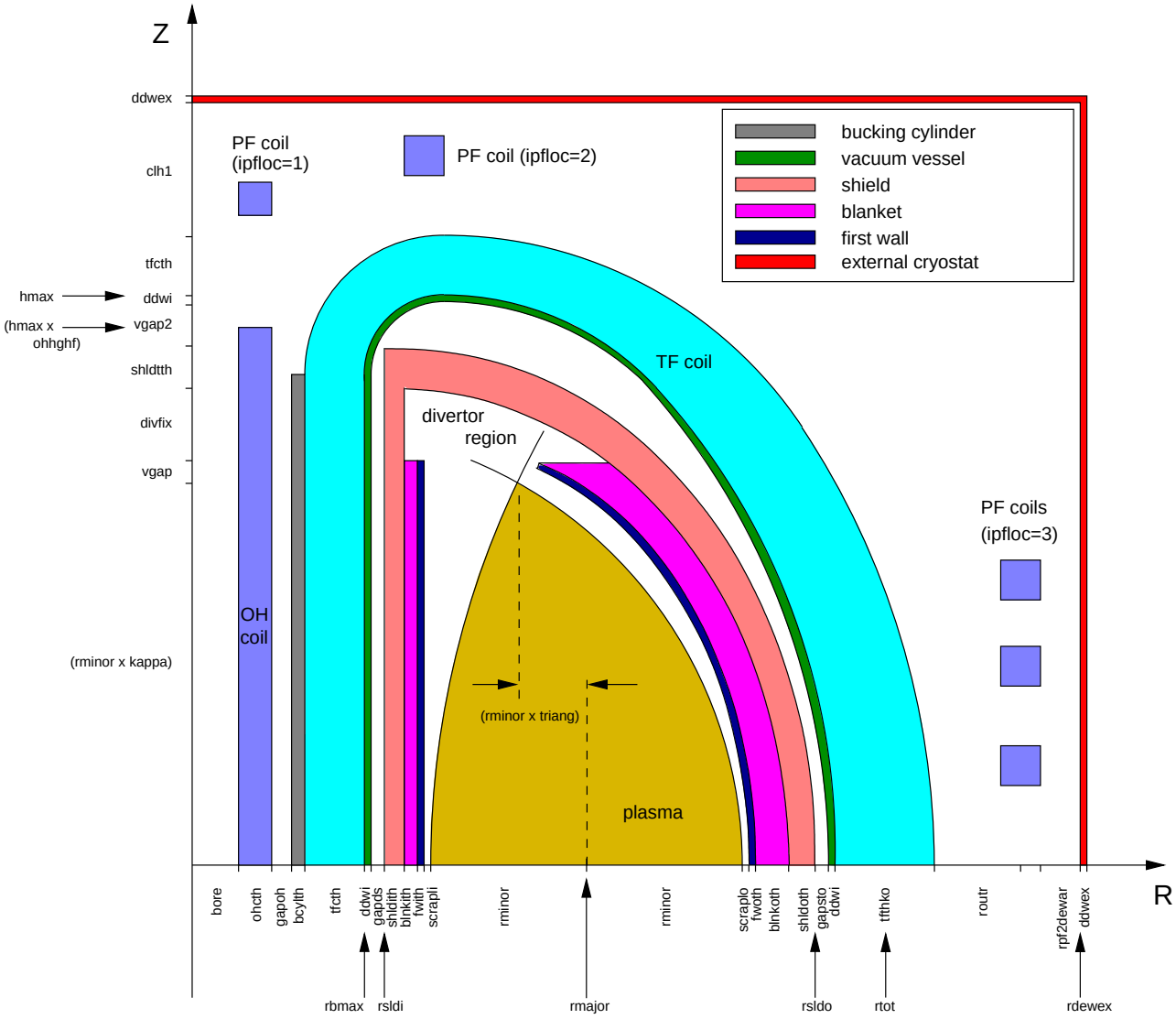

Figure 3.1: *Schematic diagram of the fusion power core of a typical tokamak power plant modelled by* ***PROCESS****, showing the relative positions of the components. A double null plasma is assumed (*`snull=0`*) — compare Figure 3.2. Also shown are the code variables used to define the thicknesses of the components. The arrowed labels adjacent to the axes are the total 'builds' to that point. The precise locations and sizes of the PF coils are calculated within the code (see Section 3.1.8).*

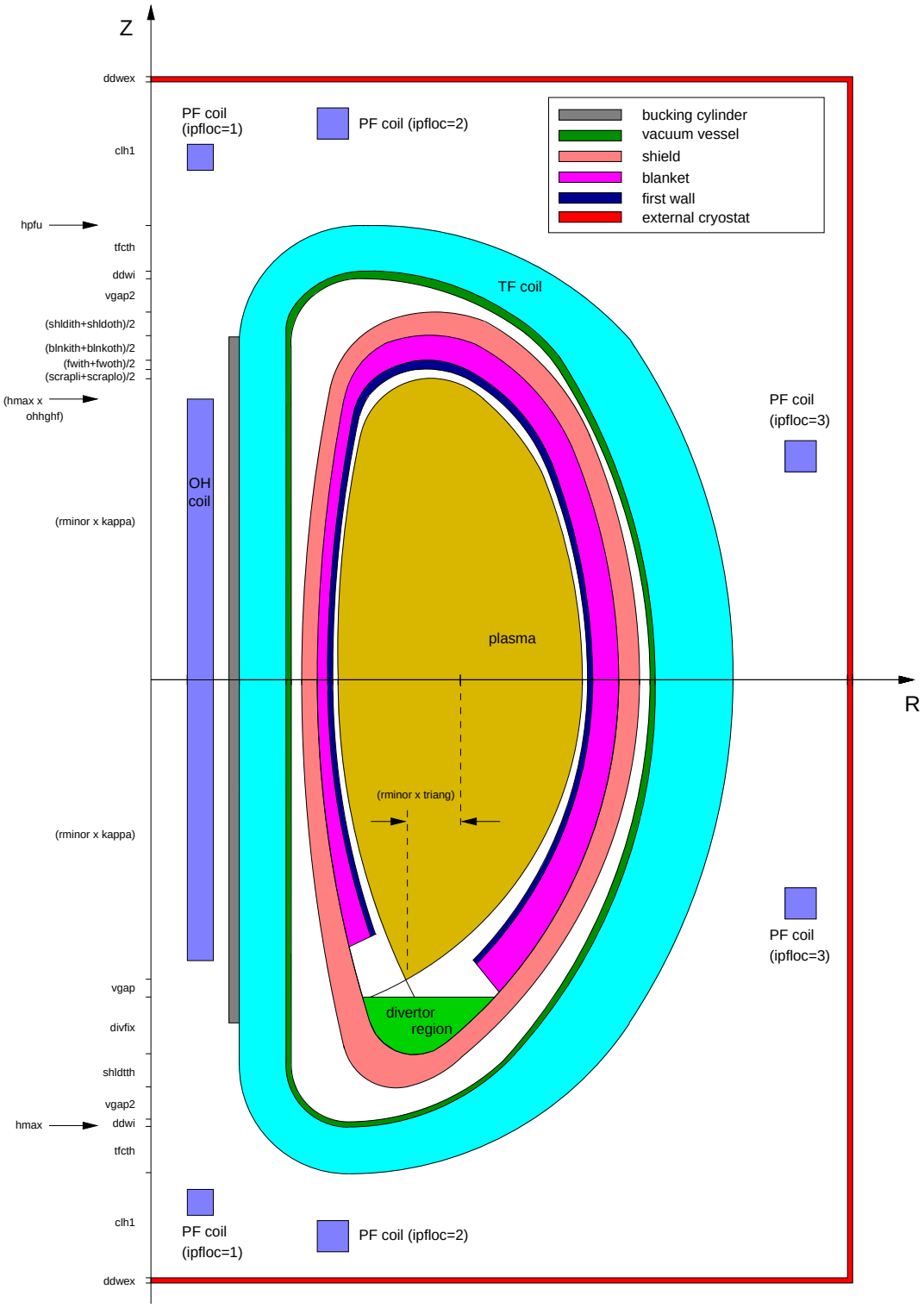

Figure 3.2: *Schematic diagram of the fusion power core of a typical tokamak power plant modelled by* `PROCESS`*, showing the relative positions of the components. A single null plasma is assumed (*`snull=1`*) — compare Figure 3.1. The radial build is the same as for a double null configuration; shown along the vertical axis are the code variables used to define the vertical thicknesses of the components. The arrowed labels adjacent to the axis are the total 'builds' (distance from the midplane, Z=0) to that point. The precise locations and sizes of the PF coils are calculated within the code (see Section 3.1.8).*

3.1.2 Plasma physics models

Arguably, the most important component of the machine is the plasma itself. By default, this is assumed to have an up-down symmetric, double null configuration (although this can be changed if desired — see Section 3.1.4), with elongation and triangularity specified by the user. A great number of physics models are coded within PROCESS to describe the behaviour of the plasma parameters such as its current, temperature, density, pressure, confinement etc., and also the various limits that define the stable operating domain.

3.1.2.1 Fusion reactions

By default, the code assumes that the deuterium-tritium (D-T) fusion reaction is utilised by the power plant being modelled:

$$\mathrm{D} + \mathrm{T} \Longrightarrow {}^4\mathrm{He} + \mathrm{n} + 17.6\,\mathrm{MeV} \tag{3.1}$$

20% of the energy produced is given to the alpha particles (^{4}He), which remain within the plasma and thermalise (slow down) due to collisions, thus heating the plasma. The remaining 80% is carried away by the neutrons, which deposit their energy within the blanket and shield. The tritium fraction of the D-T fuel is controlled using variable `ftr`.

PROCESS can also model D-^{3}He power plants, which utilise the following primary fusion reaction:

$$\mathrm{D} + {}^3\mathrm{He} \Longrightarrow {}^4\mathrm{He} + \mathrm{p} + 18.3\,\mathrm{MeV} \tag{3.2}$$

The fusion reaction rate is significantly different from that for D-T fusion, and the power flow from the plasma is modified since charged particles are produced rather than neutrons. Because only charged particles (which remain in the plasma) are produced by this reaction, the whole of the fusion power is used to heat the plasma. Useful energy is extracted from the plasma since the radiation power produced is very high, and this can be converted to electricity in a number of ways.

Since the temperature required to ignite the D-^{3}He reaction is considerably higher than that for D-T, it is necessary to take into account the following D-D reactions, which have significant reaction rates at such temperatures:

$$\mathrm{D} + \mathrm{D} \Longrightarrow {}^3\mathrm{He} + \mathrm{n} + 3.27\,\mathrm{MeV} \tag{3.3}$$

$$\mathrm{D} + \mathrm{D} \Longrightarrow \mathrm{T} + \mathrm{p} + 4.03\,\mathrm{MeV} \tag{3.4}$$

Also, as tritium is produced by the latter reaction, D-T fusion is also possible. As a result, there is still a small amount of neutron power extracted from the plasma.

D-^{3}He tokamak power plants do not include blankets, because of the near absence of neutrons leaving the plasma, and the fact that no tritium needs to be produced for fuel.

The use of the D-^{3}He coding within PROCESS is controlled using the switch `idhe3`. If `idhe3 = 1` then the D-^{3}He fusion reaction is assumed, otherwise the D-T reaction is assumed. The ratio of the "fuel" ions (D, ^{3}He and T) may be modified if `idhe3 = 1` using the three variables `fdeut`, `fhe3` and `ftrit`, respectively.

Fusion power calculations

For the case of D-T fusion reactions (`idhe3 = 0`), switch `iiter` controls which of two numerical models for the fusion power calculations should be used. If `iiter = 1`, the (1989) ITER model [14] is used, otherwise an integration method over the plasma profiles is used.

3.1.2.2 Plasma profiles

All the plasma profiles are assumed to be parabolic, i.e. they are of the form

$$\text{Density} : n(r) = n_0 \left(1 - \left(\frac{r}{a}\right)^2\right)^{\alpha_n} \tag{3.5}$$

$$\text{Temperature} : T(r) = T_0 \left(1 - \left(\frac{r}{a}\right)^2\right)^{\alpha_T} \tag{3.6}$$

$$\text{Current} : J(r) = J_0 \left(1 - \left(\frac{r}{a}\right)^2\right)^{\alpha_J} \tag{3.7}$$

where r varies from 0 to a, the plasma minor radius. This gives volume-averaged values $\langle n \rangle = n_0/(1+\alpha_n)$, and line-averaged values $\bar{n} = n_0/(1+\alpha_n)^{\frac{1}{2}}$, etc. These volume- and line-averages are used throughout the code along with the profile indices α, in the various physics models, many of which are fits to theory-based or empirical scalings. Thus, the plasma model in `PROCESS` may be described as "$\frac{1}{2}$-D". The relevant profile index variables are `alphan`, `alphat` and `alphaj`, respectively.

3.1.2.3 Beta limits

The Troyon beta limit [14, 15] is given by

$$\langle \beta \rangle < g \, \frac{I(\text{MA})}{a(\text{m})\, B_0(\text{T})} \tag{3.8}$$

where B_0 is the axial vacuum toroidal field, and β is defined with respect to the total equilibrium **B**-field [15]. The Troyon coefficient g is set using input parameter `dnbeta`. To apply the beta limit, constraint equation no. 24 should be turned on with iteration variable no. 36 (`fbetatry`). The limit can be applied to either the total plasma beta, in which case switch `iculbl` should be set to 0, to only the thermal component of the plasma beta, in which case `iculbl` should be set to 1, or to the thermal plus neutral beam components, in which case `iculbl` should be set to 2.

Aspect ratio scaling of Troyon g coefficient

Switch `gtscale` determines whether the Troyon g coefficient `dnbeta` should scale with aspect ratio (`gtscale` $\neq$ `0`), or be fixed at the input value (`gtscale = 0`).

Limiting $\epsilon.\beta_p$

To apply a limit to the value of $\epsilon.\beta_p$, where $\epsilon = a/R$ is the inverse aspect ratio, constraint equation no. 6 should be turned on with iteration variable no. 8 (`fbeta`). The limiting value of $\epsilon.\beta_p$ should be set using input parameter `epbetmax`.

3.1.2.4 Density limits

Several density limit models [15] are available in `PROCESS`. These are calculated in routine `CULDLM`, which is called by `PHYSICS`. To enforce any of these limits, turn on constraint equation no. 5 with iteration variable no. 9 (`fdene`). In addition, switch `idensl` must be set to the relevant value, as follows:-

`idensl = 1 :` ASDEX model

`idensl = 2 :` Borrass model for ITER, I

`idensl = 3 :` Borrass model for ITER, II

`idensl = 4 :` JET edge radiation model

`idensl = 5 :` JET simplified model

`idensl = 6 :` Hugill-Murakami $M.q$ model

`idensl = 7 :` Greenwald model

3.1.2.5 Plasma current scaling laws

A number of plasma current scaling laws exploiting the inverse relationship between plasma current and edge safety factor q_ψ [15] are available in `PROCESS`. These are calculated in routine `CULCUR`, which is called by `PHYSICS`. Flag `icurr` must be set to the relevant value, as follows:-

`icurr = 1 :` Peng analytic fit

`icurr = 2 :` Peng double null divertor scaling (TART)

`icurr = 3 :` Simple ITER scaling

`icurr = 4 :` Revised ITER scaling

`icurr = 5 :` Todd empirical scaling, I

`icurr = 6 :` Todd empirical scaling, II

`icurr = 7 :` Connor-Hastie model

3.1.2.6 Confinement time scaling laws

Many energy confinement time scaling laws are present within `PROCESS`, for tokamaks, RFPs or stellarators. These are calculated in routine `PCOND`. The value of `isc` determines which of the scalings is used in the plasma energy balance calculation. Table 3.1 summarises the available scaling laws.

3.1.2.7 Bootstrap current scalings

The fraction of the plasma current provided by the so-called bootstrap effect can be either input into the code directly, or calculated using one of three methods, as summarised here.

- Direct input: To input the bootstrap current fraction directly, set `bscfmax` to (-1) times the required value.
- ITER scaling [14]: To use the ITER scaling method for the bootstrap current fraction, set `ibss = 1`, and `bscfmax` to the maximum required bootstrap current fraction (≤ 1). This method is valid at high aspect ratio only.
- General scaling [16]: To use a more general scaling method, set `ibss = 2`, and `bscfmax` to the maximum required bootstrap current fraction (≤ 1).
- Numerically fitted scaling [17]: To use a numerically fitted scaling method, valid for all aspect ratios, set `ibss = 3`, and `bscfmax` to the maximum required bootstrap current fraction (≤ 1).

isc	scaling law	reference
1	Neo-Alcator (ohmic)	[14]
2	Mirnov (H-mode)	[14]
3	Merezhkin-Muhkovatov (L-mode)	[14]
4	Shimomura (H-mode)	JAERI-M 87-080 (1987)
5	Kaye-Goldston (L-mode)	Nuclear Fusion **25** (1985) p.65
6	ITER 89-P (L-mode)	Nuclear Fusion **30** (1990) p.1999
7	ITER 89-O (L-mode)	[14]
8	Rebut-Lallia (L-mode)	Plasma Physics and Controlled Nuclear Fusion Research **2** (1987) p. 187
9	Goldston (L-mode)	Plas. Phys. Controlled Fusion **26** (1984) p.87
10	T10 (L-mode)	[14]
11	JAERI-88 (L-mode)	JAERI-M 88-068 (1988)
12	Kaye-Big Complex (L-mode)	Phys. Fluids B **2** (1990) p.2926
13	ITER H90-P (H-mode)	
14	ITER Mix (minimum of 6 and 7)	
15	Riedel (L-mode)	
16	Christiansen et al. (L-mode)	JET Report JET-P (1991) 03
17	Lackner-Gottardi (L-mode)	Nuclear Fusion **30** (1990) p.767
18	Neo-Kaye (L-mode)	[14]
19	Riedel (H-mode)	
20	ITER H90-P (amended)	Nuclear Fusion **32** (1992) p.318
21	Large Helical Device (stellarator)	Nuclear Fusion **30** (1990) p.11
22	Gyro-reduced Bohm (stellarator)	Bull. Am. Phys. Society, **34** (1989) p.1964
23	Lackner-Gottardi (stellarator)	Nuclear Fusion **30** (1990) p.767
24	ITER-93H (H-mode)	Plasma Physics and Controlled Nuclear Fusion Research (Proc. 15th Int. Conf., Seville, 1994) IAEA-CN-60/E-P-3
25	TITAN (RFP)	TITAN RFP Fusion Reactor Study, Scoping Phase Report UCLA-PPG-1100, page 5–9, Jan 1987
26	ITER H-97P ELM-free (H-mode)	J. G. Cordey et al., EPS Berchtesgaden, 1997
27	ITER H-97P ELMy (H-mode)	J. G. Cordey et al., EPS Berchtesgaden, 1997
28	ITER-96P (L-mode)	Nuclear Fusion **37** (1997) p.1303
29	Valovic modified ELMy (H-mode)	
30	Kaye PPPL April 98 (L-mode)	
31	ITERH-PB98P(y) (H-mode)	
32	IPB98(y) (H-mode)	
33	IPB98(y,1) (H-mode)	
34	IPB98(y,2) (H-mode)	
35	IPB98(y,3) (H-mode)	
36	IPB98(y,4) (H-mode)	
37	ISS95 (stellarator)	Nuclear Fusion **36** (1996) p.1063
38	ISS04 (stellarator)	Nuclear Fusion **45** (2005) p.1684

Table 3.1: *Summary of the energy confinement time scaling laws in* PROCESS.

3.1.2.8 L-H power threshold scalings

Transitions from a standard confinement mode (L-mode) to an improved confinement regime (H-mode), called L-H transitions, are observed in most tokamaks. A range of scaling laws are available that provide estimates of the auxiliary power required to initiate these transitions, via extrapolations from present-day devices. PROCESS calculates these power threshold values for the scaling laws listed in Table 3.2, in routine PTHRESH.

	name	reference
(1) (2) (3)	1996 ITER scaling: nominal 1996 ITER scaling: upper bound 1996 ITER scaling: lower bound	ITER Physics Design Description Document, D. Boucher, p.2-2
(4) (5)	1997 ITER scaling, excluding elongation 1997 ITER scaling, including elongation	J. A. Snipes, ITER H-mode Threshold Database Working Group, Controlled Fusion and Plasma Physics, 24th EPS Conference, Berchtesgaden, June 1997, vol.21A, part III, p.961
(6) (7) (8)	2008 Martin scaling: nominal 2008 Martin scaling: 95% upper bound 2008 Martin scaling: 95% lower bound	Martin et al, 11th IAEA Tech. Meeting on H-mode Physics and Transport Barriers, Journal of Physics: Conference Series **123** (2008) 012033

Table 3.2: *Summary of the L-H power threshold scalings implemented in PROCESS.*

3.1.2.9 Other plasma physics options

Neo-classical correction effects

Switch ires controls whether neo-classical correction effects [19] are included in the calculation of the plasma resistance and ohmic heating power in routine POHM, which is called by routine PHYSICS. If ires = 1, these effects are included. Note that the scaling used is only valid for aspect ratios between 2.5 and 4, and it is possible for the plasma resistance to be wrongly calculated as negative if ires = 1 and the aspect ratio is too high.

Inverse quadrature in τ_E scaling laws

Switch iinvqd determines whether the energy confinement time scaling laws due to Kaye-Goldston (isc = 5) and Goldston (isc = 9) should include an inverse quadrature scaling with the Neo-Alcator result (isc = 1). A value iinvqd = 1 includes this scaling.

Scrape-off layer

The region directly outside the last closed flux surface of the core plasma is known as the scrape-off layer, and contains no structural material. Plasma entering this region is not confined and is removed by the divertor. PROCESS treats the scrape-off layer merely as a gap. Switch iscrp determines whether the scrape-off widths should be calculated as 10% of the plasma minor radius (iscrp = 0), or set equal to the input values scrapli and scraplo (iscrp = 1).

High-Z impurities

The assumed high-Z impurity affecting the plasma composition and radiation power calculations can be switched between argon (zfear = 1) and iron (zfear = 0); the latter is the default option.

[Add section on radiation power, etc.]

3.1.3 First wall

The first wall acts as a physical barrier protecting the rest of the machine from the hot plasma. Due to its hostile environment the first wall has only a short lifetime and therefore needs to be replaced regularly. Its stainless steel structure is cooled either by gaseous helium or by pressurised water, as chosen using switch `costr` – see Section 3.1.5.1.

The first wall coolant fraction is given by the value of `fwclfr`, which is calculated if a pulsed power plant is being modelled (`lpulse = 1` — see Section 3.3), and assumed to be the input value otherwise.

Wall load calculation

Switch `iwalld` determines whether the neutron wall load (power per unit area) should be calculated using the plasma surface area (`iwalld = 1`) or the first wall area (`iwalld = 2`) as the denominator.

3.1.4 Divertor

The divertor provides a means of removing plasma reaching the scrape-off layer and heavy ions that are ejected from the first wall. By default, two divertors are assumed in the `PROCESS` tokamak, placed symmetrically above and below the plasma. The principal outputs from the code are the divertor heat load, used to determine its lifetime, and its peak temperature. The divertor is cooled either by gaseous helium or by pressurised water.

Switch `snull` controls the overall plasma configuration. Setting `snull = 0` corresponds to an up-down symmetric, double null configuration (the default), while `snull = 1` assumes a single null plasma with the divertor at the bottom of the machine. The vertical build (see Figure 3.2) and PF coil current scaling algorithms take the value of this switch into account, although not the plasma geometry at present.

The Harrison-Kukushkin-Hotston divertor model [14] developed for ITER is used (except for the case of tight aspect ratio tokamaks — see later). This is appropriate for conventional aspect ratio machines, but care should be taken in inputting the divertor magnetics for this model, and projections far from the ITER CDA machine parameters are likely to be unreliable.

The divertor calculations are carried out in routines `DIVCALL` and `DIVERT`.

3.1.5 Blanket

The blanket performs a number of tasks. An incoming neutron from a deuterium-tritium (D-T) fusion reaction in the plasma loses energy in the blanket. This energy is removed by the blanket coolant and used to produce electricity. The neutron may also react with a lithium nucleus present in the blanket to produce a tritium nucleus which can be re-used as fuel. The competing requirements of heating and tritium synthesis mean that a neutron multiplier must be present, to ensure balance between tritium destruction and creation. The blanket therefore contains beryllium to fulfil this purpose. As with the first wall, the blanket has a relatively short lifetime because of the high neutron fluence. Steel and vanadium may be used as structural materials within the blanket, which is cooled either by gaseous helium or by pressurised water.

3.1.5.1 Full thermodynamic blanket model

Switch `lblnkt` determines whether the blanket is to be simulated using a full thermodynamic model [33] (`lblnkt = 1`), or simply assumed to be made up of relevant materials (see Section 3.1.5.2) in

user-defined proportions. The former model also performs a self-consistent calculation of the thermal-to-electric conversion efficiency, whereas the latter simply uses the input value `etath`.

The following switches control the details of the full thermodynamic model of the blanket.

- **Cooling channel geometry:** The value of switch `astr` determines whether the cooling channels have a circular cross-section (`astr = 1`) or an annular cross-section (`astr = 2`). The latter case is the default.
- **Cooling channel orientation:** The value of switch `estr` determines whether the cooling channels lie in the radial direction (`estr = 1`) or in the poloidal direction (`estr = 2`). The former case is the default.
- **Coolant type:** The value of switch `costr` determines the type of coolant used in the first wall, blanket and shield. If `costr = 1`, the coolant is assumed to be gaseous helium. If `costr = 2`, the coolant is assumed to be pressurised water/steam, which is the default. This switch is used whether or not the full blanket model is used, i.e. is independent of the setting of switch `lblnkt`.
- **Boundary condition:** The value of switch `bstr` determines whether the coolant output temperature is to be fixed (`bstr = 1`) or whether the maximum blanket temperature is to be fixed (`bstr = 2`). The former case is the default. The desired coolant output temperature for `bstr = 1` is set using input parameter `xtfo`, and the required maximum blanket temperature is set using input parameter `xtb`.

3.1.5.2 Blanket materials

Table 3.3 summarises the possible options for the blanket materials. Switch `smstr` determines whether a solid blanket of Li_2O-Be (`smstr = 1`), or a liquid blanket of LiPb-Li (`smstr = 2`) is used. The former is the default, and is the type assumed if `lblnkt` $\neq$ `1`.

material	`lblnkt` $\neq$ `1`: simple model	`lblnkt = 1`: full thermodynamic model	
		`smstr=1`: solid blanket	`smstr=2`: liquid blanket
stainless steel	`fblss`	`fblss`	`fblss`
vanadium	`fblvd`	`fblvd`	`fblvd`
Li_2O	`fblli2o`	`fblli2o`	—
beryllium	`fblbe`	`fblbe`	—
LiPb	—	—	`fbllipb`
lithium	—	—	`fblli`
coolant	`vfblkt`	`vfblkt`	`vfblkt`

Table 3.3: *Summary of the materials comprising the blanket in the various scenarios available. The fractions given are all available to be modified, and should of course add up to 1.0 for any given model. The type of coolant used is given by the value of switch* `costr`.

3.1.6 Shield

The stainless steel shield reduces the neutron flux reaching the TF coils and beyond. This minimises the radiological impact of the neutrons, and their heating of the TF coils which, if superconducting, need to remain at liquid helium temperatures. The shield is cooled either by gaseous helium or by pressurised water, as chosen using switch `costr` – see Section 3.1.5.1, and as with the blanket the

energy deposited in the coolant is used to produce electricity. The shield coolant fraction is stored in input parameter `vfshld`.

3.1.7 Toroidal field coils

The toroidal field (TF) coils can be either resistive or superconducting. Switch `itfsup` should be set to `1` for superconducting coils, or `0` for purely copper coils. In the superconductor model, the *CICC* (Conductor In Cable Conduit) structure shown in Figure 3.3 is assumed, and the coils are cooled using a liquid helium cryogenic system. Among the TF coil parameters calculated by the code are the maximum allowable current density, the stresses on the structure, the energy stored and the magnetic field produced by the coils.

Each TF coil is defined in the (R, Z) plane by four circular arcs of different radius, which create a D-shaped profile. Because of the finite number of TF coils used in a tokamak (typically around 20), the toroidal field has a ripple introduced into it, the amplitude of which can be limited to a few percent by the code by adjusting the outboard gap thickness (labelled `gapsto` in Figure 3.1). Ports are often necessary for auxiliary power systems etc., and the gaps between adjacent TF coils can be made large enough to accommodate such equipment, using constraint equation no. 20 with iteration variable no. 33 (`fportsz`).

The following options are available within the superconducting TF coil model (`itfsup = 1`).

3.1.7.1 Superconducting materials

Switch `isumattf` specifies which superconducting material is to be used:

`isumattf = 1 :` Nb_3Sn superconductor, ITER critical surface parameterization [4], standard critical values

`isumattf = 2 :` (not used)

`isumattf = 3 :` NbTi superconductor

`isumattf = 4 :` Nb_3Sn superconductor, ITER critical surface parameterization [4], user-defined critical parameters

The fraction of copper present in the superconducting filaments is given by the value of variable `fcutfsu` (iteration variable no. 59). For `isumattf = 4`, important superconductor properties may be input by the user as follows: the upper critical field at zero temperature and strain is set using input parameter `bcritsc`, and the critical temperature at zero field and strain is set using input parameter `tcritsc`.

3.1.7.2 Stress model

Switch `itfmod` controls whether a simple stress model (`itfmod = 0`) or more complex stress model (`itfmod = 1`) should be used. To enforce the stress limits calculated using either of these models, constraint equation no. 31 (case stress) and/or constraint equation no. 32 (conduit stress) should be turned on with iteration variables no. 48 (`fstrcase`) and/or no. 49 (`fstrcond`), respectively. The stress limit can be adjusted using input parameters `csutf` and `csytf`.

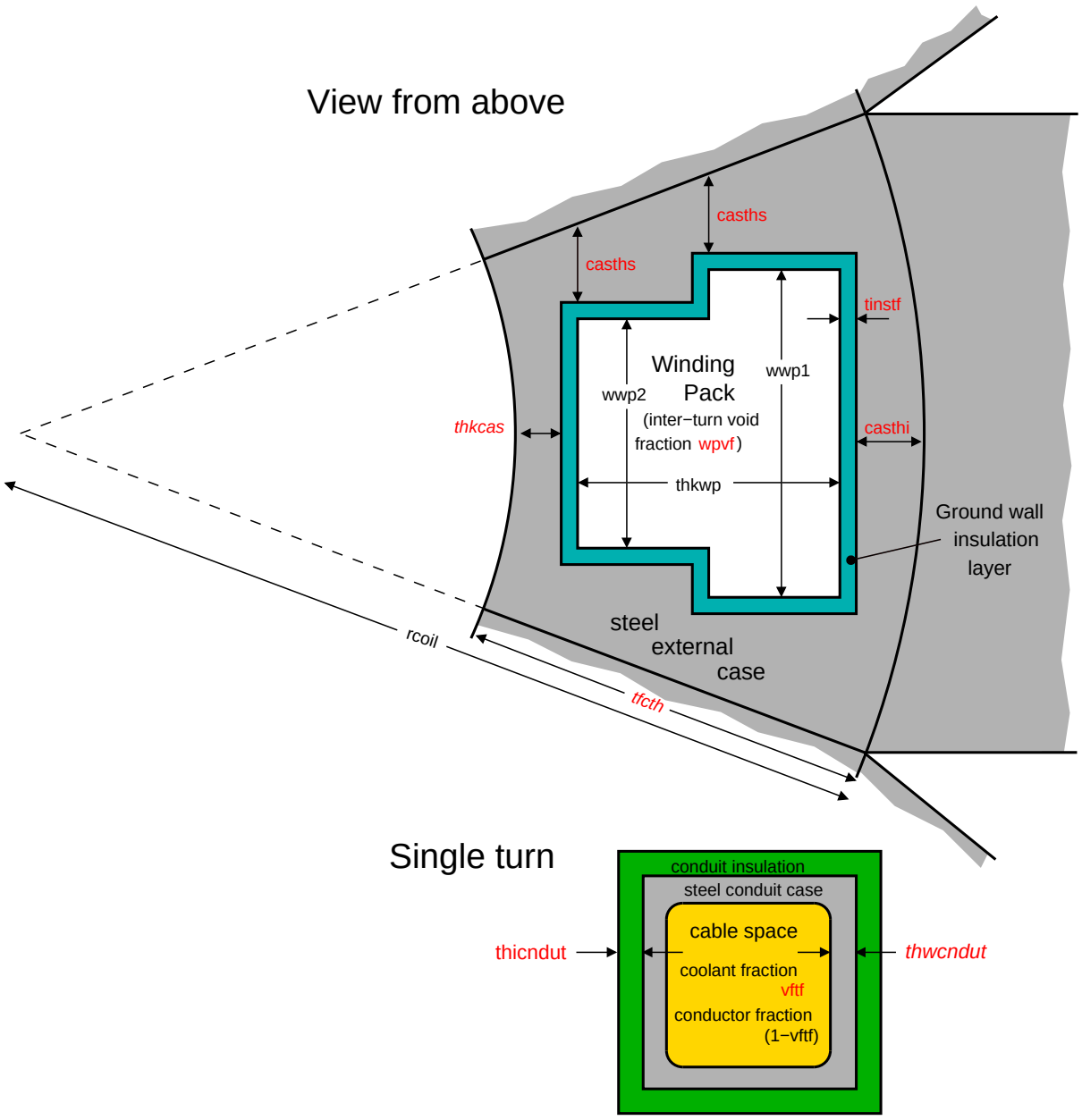

Figure 3.3: *Schematic diagram of the cross-section of the inboard leg of a superconducting TF coil, showing the CICC (Conductor In Cable Conduit) construction. The winding pack contains many turns of cable conduit. The cable space contains the superconducting filaments, and circulating liquid helium coolant. The variables shown in red may be changed by the user, and those in italics may be chosen as iteration variables.*

3.1.7.3 Current density limits

The current in the TF coils must be sufficient to produce the required toroidal field at the centre of the plasma. The field falls off at a rate $1/R$, with the peak value occurring at the outer edge of the inboard portion of the TF coil ($R_{\text{max TF}}$ = `rbmax`). The maximum TF coil current depends on the field it produces and the allowable current density.

Two limits can be applied to the current density J in the (superconducting) TF coils. To ensure that J does not exceed the critical value, constraint equation no. 33 should be turned on with iteration variable no. 50 (`fiooic`). To ensure that J does not exceed the current density protection limit, constraint equation no. 35 should be turned on with iteration variable no. 53 (`fjprot`).

(A similar constraint on the TF coil current density exists if resistive coils are being used (`itfsup = 0`). In this case, constraint equation no. 23 should be turned on with iteration variable no. 28 (`fjtfc`).)

3.1.8 Poloidal field coils

The poloidal field (PF) coils are used initially to cancel the vertical field produced at the centre of the plasma by the OH coil (Section 3.1.9) during start-up, and then to maintain the plasma position and shape during the flat-top period.

3.1.8.1 PF coil positions

The positions and sizes of the PF coils are partly input, and partly calculated after consideration of the required currents and allowable current density.

The PF coil locations are controlled using a set of switches stored in array `ipfloc` (see Figure 3.1), and are calculated in routine `PFCOIL`. The coils are (usually) organised into groups containing two PF coils placed symmetrically above and below the midplane, and each group `j` has an element `ipfloc(j)` assigned to it. Input parameter `ngrp` should be set to the number of groups, and `ncls(j)` should be assigned the number of coils in each group — which should be `2` in each case.

In the following, all variables are defined in the variable descriptor file `vardes.html`. The values for `rpf1`, `rpf2`, `zref(j)` and `routr` should be adjusted by the user to locate the PF coils accurately.

The three possible values of `ipfloc(j)` correspond to the following PF coil positions:

`ipfloc(j) = 1 :` PF coils are placed above the OH coil (one group only);

$$\begin{aligned} R &= \texttt{rohc} + \texttt{rpf1} \\ Z &= \pm(\texttt{hmax} * \texttt{ohhghf} + 0.1 + 0.5 * (\texttt{hmax} * (1.0\texttt{D0} - \texttt{ohhghf}) + \texttt{tfcth} + 0.1)) \end{aligned}$$

`ipfloc(j) = 2 :` PF coils are placed above the TF coils (one group only);

$$\begin{aligned} R &= \texttt{rmajor} + \texttt{rpf2} * \texttt{triang} * \texttt{rminor} \\ Z &= \pm(\texttt{hmax} + \texttt{tfcth} + 0.86) \end{aligned}$$

`ipfloc(j) = 3 :` PF coils are placed radially outside the TF coils (any number of groups (?));

$$\begin{aligned} R &= \texttt{rtot} + \texttt{tfthko}/2.0\texttt{D0} + \texttt{routr} \\ Z &= \pm(\texttt{rminor} * \texttt{zref(j)}) \end{aligned}$$

3.1.8.2 PF coil currents

The PF coil currents are calculated in routine `EFC`, and vary as a function of time during the tokamak operation as indicated in Figure 3.4.

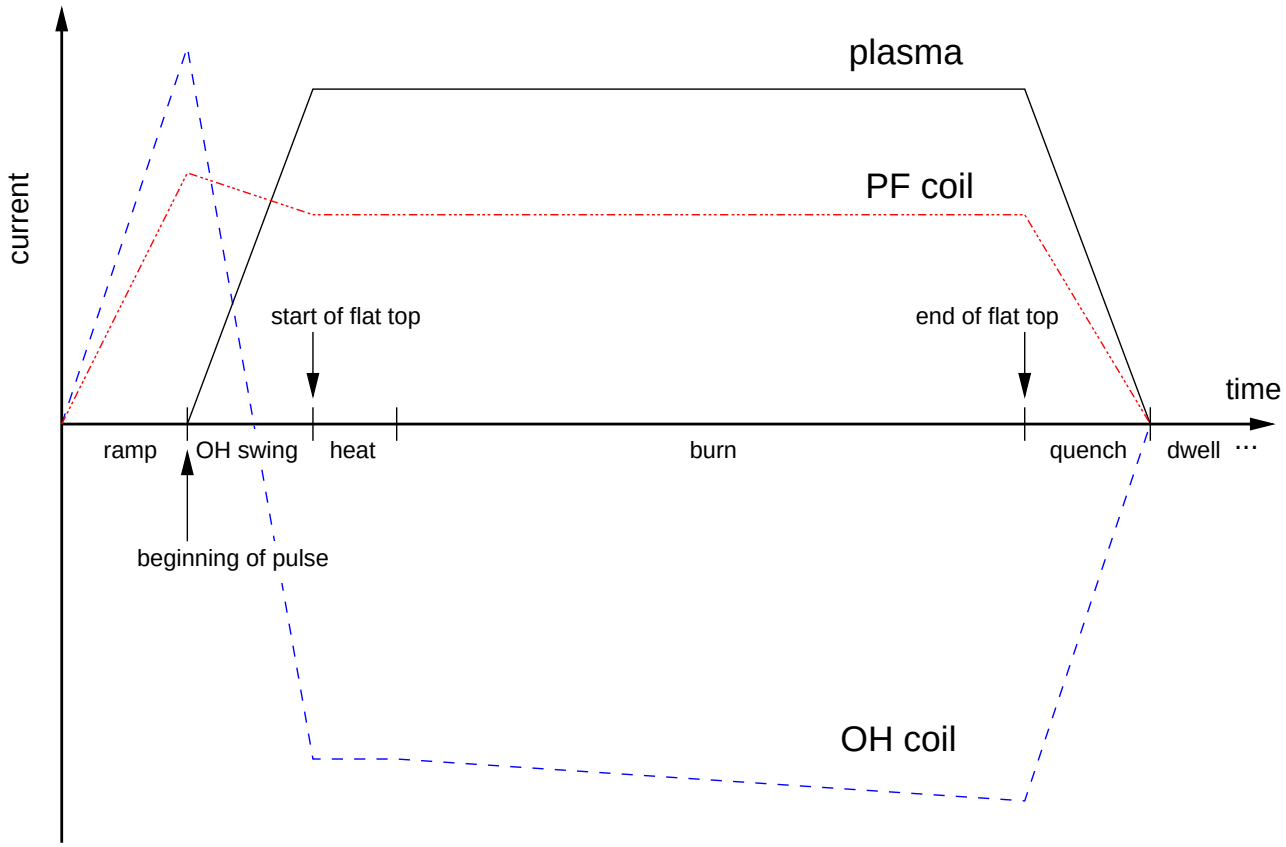

Figure 3.4: *Plot showing schematically the current waveforms for the plasma, a typical PF coil, and the OH coil.*

3.1.8.3 PF coil resistance

The PF coils can be either resistive or superconducting. This is determined from the value of `ipfres`. If `ipfres = 0`, the PF and OH coils are assumed to be superconducting. If `ipfres = 1`, they are assumed to be resistive, with their resistivity given by the value of variable `pfclres`.

3.1.8.4 Superconducting materials

If `ipfres = 0`, switch `isumatpf` specifies which superconducting material is to be used for the PF and OH coils:

`isumatpf = 1 :` binary Nb_3Sn superconductor

`isumatpf = 2 :` ternary Nb_3Sn superconductor

`isumatpf = 3 :` NbTi superconductor

3.1.9 Ohmic heating coil

The ohmic heating (OH) coil is a PF coil used primarily during start-up (but also during the burn phase) to create and maintain the plasma current by inductive means. Swinging (changing) the current through the OH coil causes a change in the flux linked to the plasma region, inducing a current in it. PROCESS calculates the amount of flux required to produce the plasma current, and also the amount actually available. The code measures the magnetic flux in units of Volt-seconds (= Webers). The OH coil is sometimes referred to as the central solenoid, and can be either resistive or superconducting (controlled via switches `ipfres` and `isumatpf` as for the other PF coils).

Switch `iohcl` controls whether an OH coil is present. A value of `1` denotes that this coil is present, and should be assigned a non-zero thickness `ohcth`. A value of `iohcl = 0` denotes that no OH coil is present, in which case the thickness `ohcth` should be set to zero. No PF coils should be located at positions defined by `ipfloc(j) = 1` if no OH coil is present.

3.1.9.1 OH coil swing time

In the steady-state power plant scenario (`lpulse` $\neq$ `1` — see Section 3.3), the length of time taken for the OH coil current to reverse (see Figure 3.4) is determined from the value of switch `tohsin`. If `tohsin = 0.0D0`, then the swing time `tohs` in seconds is given by `tohs` $= I_p/0.5$, where I_p is the plasma current in MA. Furthermore, the PF coil ramp time `tramp` and shutdown time `tqnch` are set equal to `tohs`. If `tohsin` $\neq$ `0.0D0`, the swing time `tohs = tohsin`, and the ramp and shutdown times are input parameters.

If, however, a pulsed power plant is being modelled (`lpulse = 1`), the OH coil swing time `tohs` is either an input parameter, or it can be iterated by using iteration variable 65. The ramp and shutdown times in the pulsed case are always set equal to `tohs`. To ensure that the plasma current ramp rate during start-up is prevented from being too high, as governed by the requirement to maintain plasma stability in l_i-q_ψ space, constraint equation no. 41 should be turned on with iteration variable no. 66 (`ftohs`).

3.1.9.2 Current density limits

The current density in the OH coil can be limited at the beginning-of-pulse (BOP) and at the end-of-flat-top (EOF — see Figure 3.4). The limiting value is dependent on the maximum allowable stress in the coil as given by the value of `sigpfalw`. To limit the current density at the BOP, constraint equation no. 27 should be turned on with iteration variable no. 39 (`fjohc0`). To limit the current density at the EOF, constraint equation no. 26 should be turned on with iteration variable no. 38 (`fjohc`).

3.1.10 Auxiliary power systems: heating and current drive

3.1.10.1 Current Drive

The use of purely inductive current drive leads to pulsed plant operation because of the limited flux swing that can be achieved using the OH coil. This poses problems due to the fact that fatigue failures may result, and there would also be a need for thermal storage to maintain a level supply between pulses. However, the plasma current can also be produced and maintained (partially or wholly) using non-inductive means which, in principle, removes this restriction. PROCESS contains a number of auxiliary current drive schemes, including various RF methods (Lower Hybrid, Electron Cyclotron,

and Ion Cyclotron (Fast Wave) current drives) and also Neutral Beam current drive systems. The code calculates the efficiency and the resulting power requirements of the chosen system.

The fraction of the required flux swing (Volt-seconds) to be produced by non-inductive means, `fvsbrnni`, should be set, and flag `irfcd` should be set to `0` for purely inductive scenarios, or `1` otherwise. The current drive efficiency model to be used in this latter case is defined by the value of switch `iefrf`:-

`iefrf = 1 :` Fenstermacher Lower Hybrid model

`iefrf = 2 :` Ion cyclotron model [14]

`iefrf = 3 :` Fenstermacher electron cyclotron resonance model

`iefrf = 4 :` Ehst Lower Hybrid model

`iefrf = 5 :` ITER neutral beam model [14, 15]

`iefrf = 6 :` Culham Lower Hybrid model [15]

`iefrf = 7 :` Culham electron cyclotron model [15]

`iefrf = 8 :` Culham neutral beam model [15]

`iefrf = 9 :` Oscillating Field current drive (RFPs only — see Section 3.6.1.5)

It is sometimes useful to adjust artificially the current drive efficiency values produced by these routines. This can be achieved by setting the scaling coefficient `feffcd`. The wall plug to plasma efficiencies can also be adjusted, by changing the relevant variable (`etaech`, `etalh`, `etanbi` or `etaof`).

3.1.10.2 Plasma heating

In addition to current drive, some auxiliary power can be used purely to heat the plasma. The value of input parameter `pheat` determines the amount of auxiliary *heating* power (in Watts) to be applied to the plasma. This variable may be used as an iteration variable (no. 11).

3.1.10.3 Ignited plasma

Switch `ignite` can be used to denote whether the plasma is ignited, i.e. fully self-sustaining without the need for any injected auxiliary power during the burn. If `ignite = 1`, the calculated injected power does not contribute to the plasma power balance, although the cost of the auxiliary power system is taken into account (the system is then assumed to be required to provide heating etc. during the plasma start-up phase only — use `pheat` to indicate the power requirement). If `ignite = 0`, the plasma is not ignited, and the auxiliary power is taken into account in the plasma power balance during the burn phase.

3.1.11 Structural components

Structural components are required to provide support for the fusion power core systems against gravity and the magnetic forces that will be encountered during operation. The required structural masses and their costs are calculated.

3.1.11.1 Bucking cylinder

The bucking cylinder provides some strength to the inboard TF coil structure. If the TF coils are superconducting, the bucking cylinder is cooled by the cryogenic system.

3.1.12 Power conversion and heat dissipation systems

The PROCESS power plant takes into account all the systems required to perform the necessary conversion of fusion power to electricity, from the coolant systems in the plant components to the heat exchangers and turbines. Figure 3.5 shows schematically the overall power transfer mechanisms used by the code.

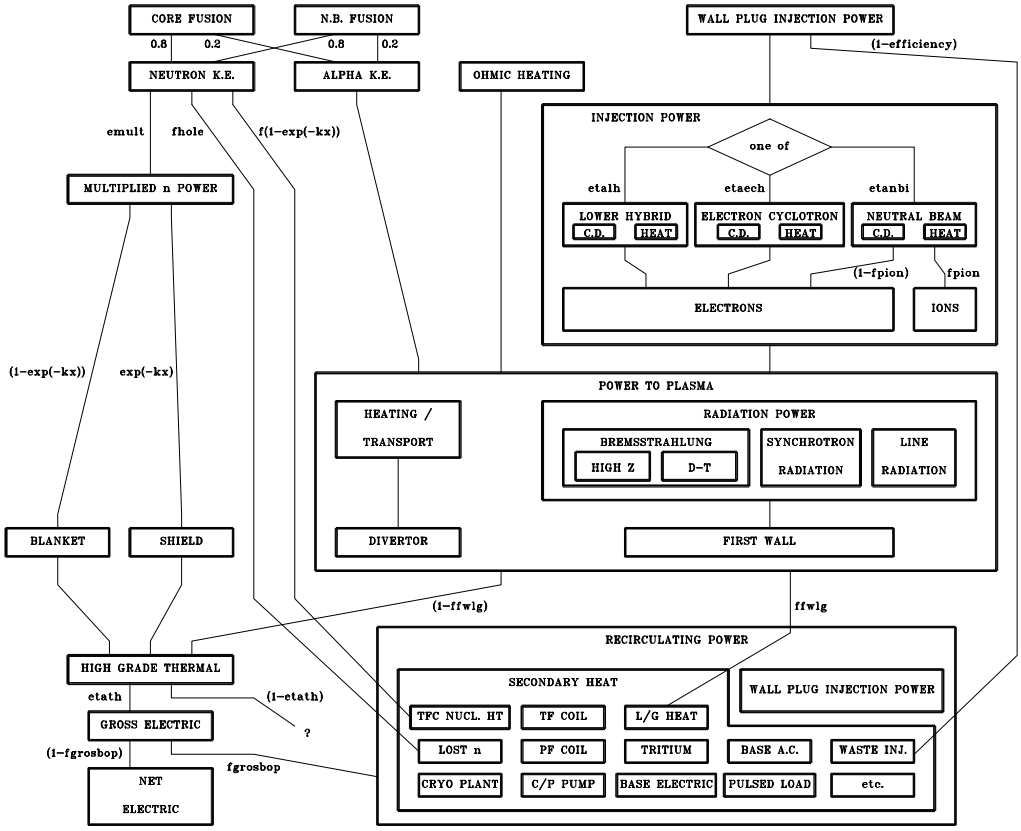

Figure 3.5: *Schematic diagram showing the power conversion mechanisms used in PROCESS[3, Note 0166].*

Many of the power conversion efficiencies shown in Figure 3.5 can be adjusted by the user.

3.1.13 Cryostat and vacuum system

The internal vacuum vessel provides a toroidal evacuated chamber containing the plasma, first wall, blanket and shield, and the space between this item and the external cylindrical cryostat encloses those components that need to operate at liquid helium temperatures. These include any superconducting (TF or PF) coils, the inter-coil structure and the bucking cylinder. PROCESS calculates the cryogenic power load and the resulting heat exchanger requirements.

The vacuum system is used for four different processes. Firstly, before plasma operations the chamber must be evacuated to remove outgassed impurities from the structure. Secondly, the chamber must be re-evacuated between burn operations. Thirdly, helium ash must be removed to prevent it from diluting the fuel. Finally, deuterium and tritium is removed on a steady state basis. PROCESS calculates the parameters of a vacuum system that satisfy all four requirements, with the option of either turbo pumps or cryo pumps being used.

Switch `ntype` controls whether a turbopump (`ntype = 0`) or a cryopump (`ntype = 1`) is used in the vacuum system.

3.1.14 Buildings

The volume and ground area of all the various buildings on a power plant site are included in the PROCESS calculations for the benefit of the costing algorithms.

3.2 Tight Aspect Ratio Tokamak Model

PROCESS has the ability to perform studies on tokamaks in the low aspect ratio regime (major radius $\leq 2\times$ minor radius). The physics and engineering issues [5] associated with these machines are somewhat different from those of conventional aspect ratio, and this is reflected by the following special models [2] in PROCESS.

1. The inboard build of a tight aspect ratio tokamak (TART) is very different from that in a conventional tokamak. There is no inboard blanket (and possibly no inboard shield), and the inboard TF coil legs are replaced by a single centrepost. The radial build is altered so that, starting from the centreline ($R = 0$), the component order is: bucking cylinder, TF coil, gap, OH coil, cryostat, and then continuing as in Figure 3.1.

2. Tight aspect ratio tokamaks have resistive TF coils that combine into a single centrepost at the centre of the machine. The centrepost is constructed from copper (as are the outboard TF coil sections), and is tapered lengthways so that it is narrowest at the midplane of the device. Routine CNTRPST calculates various parameters relevant to the centrepost, including the pump pressure, maximum temperature and pipe radius, and these may be limited using constraint equations 43 to 46 if required:

   - Equation 43 is a consistency equation for the average centrepost temperature.
   - Equation 44 can be used to limit the peak centrepost temperature to a maximum value (`ptempalw`) using iteration variable no. 68 (`fptemp`).
   - Equation 45 can be used to force a lower limit to the edge safety factor, using iteration variable no. 71 (`fq`).
   - Equation 46 can be used to apply an upper limit to the ratio of plasma current to TF coil ("rod") current, using iteration variable no. 72 (`fipir`).

3. A gaseous divertor model is used, and a simple divertor heat load calculation is employed, rather than the ITER-CDA like divertor assumed for conventional aspect ratio tokamaks.

4. A simple PF coil current scaling algorithm is available for use with the TART option.

5. The plasma shaping terms (elongation and triangularity) can be calculated directly given the aspect ratio. Switch `ishape` controls whether the input values for the plasma elongation (`kappa`) and triangularity (`triang`) should be used (`ishape = 0`), or whether they should be scaled with the plasma aspect ratio (`ishape = 1`). The latter case should only be used with a TART machine.

6. Among the physics models that differ from those relevant to conventional aspect ratio machines are (i) the bootstrap current fraction, (ii) the Troyon beta limit, and (iii) the neutron heating of the centrepost.

3.2.1 Tight aspect ratio tokamak switches

Switch `itart` provides overall control of the TART switches within the code, and subroutine `CHECK` ensures that no conflicting values are inadvertently set by the user in the input file. Table 3.4 summarises the switch values relevant to each aspect ratio regime.

switch	conventional aspect ratio `itart = 0`	tight aspect ratio `itart = 1`
`ishape`	0	0, 1
`ibss` (Section 3.1.2.7)	1, 2, 3	2, 3
`icurr` (Section 3.1.2.5)	1, 3, 4, 5, 6, 7	2
`itfsup` (Section 3.1.7)	0, 1	0

Table 3.4: *Summary of the switch values in PROCESS that relate to conventional aspect ratio and tight aspect ratio machines.*

3.3 Pulsed Plant Operation

If the plasma current is not to be driven by purely non-inductive means, it is necessary to operate the plant in a pulsed manner as the current swing in the OH/PF coils cannot be continued indefinitely. `PROCESS` can perform a number of calculations relevant to a pulsed power plant, as detailed below.

Switch `lpulse` determines whether the power plant is assumed to be based on steady-state (`lpulse = 0`) or pulsed (`lpulse = 1`) operation.

3.3.1 Thermal cycling package

This performs calculations on the first wall of the machine. Evaluation of the mechanical and thermal stresses on this component lead to a measure of the maximum number of cycles to which the first wall can be subjected, and hence to the minimum allowable length of each reactor cycle for a specified first wall lifetime. The cycle time can be constrained to be at least the minimum value by turning on constraint equation no. 42 with iteration variable no. 67 (`ftcycl`).

The thickness of the first wall is constrained to lie within lower and upper bounds, which ensures that it can withstand the internal coolant pressure and the peak temperature and neutron fluence.

Switch `itcycl` activates the desired model for the first wall axial stress calculations. If `itcycl = 1` (the default), the wall is fully constrained axially, and no bending can occur. If `itcycl = 2`, there is no constraint on the axial motion, but no bending can occur. Finally, if `itcycl = 3`, again there is no axial constraint, and bending is allowed to occur.

3.3.2 First wall coolant temperature rise limit

The rise in temperature of the first wall coolant can be limited to be no more than the value of `dtmpmx` by turning on constraint equation no. 38 with iteration variable no. 62 (`fdtmp`).

3.3.3 First wall peak temperature limit

The maximum first wall temperature can be limited to be no more than the value of variable `tpkmax` by turning on constraint equation no. 39 with iteration variable no. 63 (`ftpeak`).

3.3.4 Start-up power requirements

The minimum auxiliary power required during the start-up (ignition) phase is calculated on the basis of a POPCON analysis. Ignition is accessed via the so-called Cordey Pass (the path in plasma density–temperature space which minimises the power requirement) and the code ensures that there is sufficient auxiliary power to accommodate this. In fact, this calculation is very CPU-intensive, so the relevant routine is not called at present. In practice, the auxiliary power tends to exceed the minimum allowable value anyway, without any need to constrain it to do so.

The auxiliary power reaching the plasma can be forced to be more than the minimum allowable value `auxmin` by turning on constraint equation no. 40 with iteration variable no. 64 (`fauxmn`). The value of `auxmin` is determined by the code if the start-up model is activated, otherwise it may be initialised via the input file.

3.3.5 OH coil swing time

This calculation ensures that the plasma current ramp rate during start-up is prevented from being too high, as governed by the requirement to maintain plasma stability in l_i - q_ψ space (see Section 3.1.9.1).

3.3.6 Burn time

The length of the burn time is calculated from the surplus volt-seconds available from the OH/PF coil system during the plasma burn phase, after the flux required during the plasma start-up is taken into account. A minimum burn time can be enforced via constraint equation no. 13 and iteration variable no. 21 (`ftburn`).

3.3.7 Thermal storage

During every cycle there is a period when no fusion power is produced. The net electric output from the plant must, however, be maintained, and this is achieved using thermal storage. There are three

types of thermal storage available within PROCESS, and the value of switch `istore` determines which is to be used. If `istore = 1` (the default), option 1 of Ref. [34] is assumed, which utilises the thermal storage inherent in the machine's steam cycle equipment. This should be used if the machine down time is less than 100 seconds. If `istore = 2`, option 2 of Ref. [34] is assumed, which uses the same method as before, but augments it with an additional boiler. This may be used for machine down times of up to 300 seconds. Finally, if `istore = 3`, a large stainless steel block acts as the thermal storage medium.

3.4 Hydrogen Production Facility

Fusion power plants have been mooted as a means of producing hydrogen for use in fuel cells for cars, for instance. PROCESS includes options to enable the plant to produce hydrogen using a number of different processes.

To include the production of hydrogen by the power plant, it is necessary to set the switch `ihplant`, as follows:

`ihplant = 0 :` No hydrogen production (default)

`ihplant = 1 :` Hydrogen production by low temperature electrolysis

`ihplant = 2 :` Hydrogen production by endothermic high temperature electrolysis

`ihplant = 3 :` Hydrogen production by exothermic high temperature electrolysis

`ihplant = 4 :` Hydrogen production by thermo-chemical processes

Table 3.5 describes the additional options available for each of the types of hydrogen production given above. The different processes use either electrical power or thermal power directly, so the required inputs differ. Variable `helecmw` (iteration variable no. 87) is the electrical power in MW required for hydrogen production, while `hthermmw` (iteration variable no. 88) is the thermal power required. Note that `hthermmw` must not be used as an iteration variable if `ihplant` $\neq 4$, as it will be calculated from the required electrical power instead. Similarly, `helecmw` must not be used as an iteration variable if `ihplant = 4`. The efficiency variables given in Table 3.5 are all input parameters, and are the factors

hydrogen plant option	`helecmw`	`hthermmw`	efficiency variable
`ihplant=1`	input	zero	`etahlte`
`ihplant=2`	input	calculated	`etahten`
`ihplant=3`	input	calculated	`etahtex`
`ihplant=4`	zero	input	`etahth`

Table 3.5: *Summary of the variables in PROCESS that relate to the different hydrogen plant processes.*

to be used to convert the value of `helecmw` to the amount of hydrogen produced (in MW equivalent); these can be greater than unity in all cases except `ihplant = 4`.

3.5 Stellarator Model

The code has the ability to perform calculations based on the physics and engineering of a stellarator, which, although being a toroidal device, is radically different in a number of ways from a tokamak.

The model is derived from two main sources; the physics is based on that assumed by the U.S. stellarator reactor study group [6], and the coil set is scaled from the proposed Wendelstein VII-X design, a modular advanced stellarator with a 5-period Helias configuration [7].

To activate the stellarator coding, it is necessary to create a file `device.dat`, containing the single character `1` in the first row, in the working directory (see Section 4.1). This has the effect of setting the internally-used switch `istell = 1`. If the file is absent, or its first character is set to something other than `1`, the stellarator model is not used, and `istell` is set to `0`.

The stellarator model is largely contained within source file `stellarator.f90`. The consistency equations relevant for tokamaks (see Section 2.4) should be used without modification, to ensure that the coil currents and the fields they produce are consistent with the plasma parameters. An additional consistency equation (`17`) should be used to ensure that the radial build is correct (see Section 3.5.2.1).

3.5.1 Stellarator physics

Much of the physics is identical to that for tokamaks, including the plasma composition, fusion power considerations and energy conversion and transport within the plasma. However, some physics topics do differ between stellarators and tokamaks, as follows.

3.5.1.1 Absense of plasma current

Stellarators have zero plasma current, so no current scalings are required.

3.5.1.2 Beta limit

The beta limit is assumed to be 5%, based on 3-D MHD calculations [20]. To apply the beta limit, constraint equation no. 24 should be turned on with iteration variable no. 36 (`fbetatry`).

3.5.1.3 Density limit

The density limit relevant to stellarators has been proposed to be [21]

$$n_{\max} = 0.25 \left(P B_0 / R_0 \, a_p^2 \right)^{\frac{1}{2}} \tag{3.9}$$

where n is the line-averaged electron density in units of 10^{20} m^{-3}, P is the absorbed heating power (MW), B_0 is the on-axis field (T), R_0 is the major radius (m), and a_p is the plasma minor radius (m). To enforce the density limit, turn on constraint equation no. 5 with iteration variable no. 9 (`fdene`).

3.5.1.4 τ_E scaling laws

Five confinement time scaling laws relevant to stellarators are present within `PROCESS`. The value of switch `isc` determines which of these is used in the plasma energy balance calculation.

$$\tau_E \text{ (Large Helical Device: } \texttt{isc=21}) = 0.17 \, R_0^{0.75} \, a_p^2 \, n^{0.69} \, B_0^{0.84} \, P^{-0.58} \tag{3.10}$$

$$\tau_E \text{ (Gyro-reduced Bohm: } \texttt{isc=22}) = 0.25 \, B_0^{0.8} \, n^{0.6} \, P^{-0.6} \, a_p^{2.4} \, R_0^{0.6} \tag{3.11}$$

$$\tau_E \text{ (Lackner-Gottardi: } \texttt{isc=23}) = 0.17 \, R_0 \, a_p^2 \, n^{0.6} \, B_0^{0.8} \, P^{-0.6} \, \iota^{0.4} \tag{3.12}$$

Here, $\bar{\iota}$ is the rotational transform, which is equal to the reciprocal of the tokamak safety factor q.

3.5.1.5 Heating power options

Stellarators require no current drive, although provision for auxiliary heating does need to be present. The method by which auxiliary heating power is supplied is determined by the switch `isthtr`:

`isthtr = 1 :` electron cyclotron resonance heating

`isthtr = 2 :` lower hybrid heating

`isthtr = 3 :` neutral beam injection

The value of variable `pheat` determines the actual amount of auxiliary heating power (in Watts) to be applied to the plasma. This variable may be used as an iteration variable (no. 11). Switch `ignite` may be used if necessary — see Section 3.1.10.3.

3.5.2 Machine configuration

There are a large number of possible stellarator configurations. The one chosen for the `PROCESS` model is based on the **HELI**cal **A**dvanced **S**tellarator (Helias) concept, in which all the coils resemble distorted, non-planar TF coils — no helical coils or tokamak-like PF coils are present. This approach was chosen because at the time the model was introduced into the code the Helias was the most promising stellarator concept for a power plant, with a modular engineering design and optimised plasma, MHD and magnetic field properties [8]. Furthermore, this choice also enabled the coil engineering issues to be coded easily by simply modifying the existing `PROCESS` superconducting TF coil model. The coil geometry is scaled from the Wendelstein VII-X device, which is based on a five field-period Helias configuration.

3.5.2.1 Machine build

Since a stellarator is inherently non-axisymmetric, the build of the `PROCESS` stellarator is defined in terms of the mean thicknesses of components. This allows the code to use the existing algorithms for the surface areas, volumes and masses of the plasma and the machine's structural materials, without introducing unacceptably large errors in the calculated values.

It is important that the relative size of the plasma and coils within the stellarator being modelled remains correct. For the Helias configuration used in `PROCESS` the average coil minor radius is 2.6 times the average plasma minor radius. This is enforced in the code by using constraint equation no. 17.

Furthermore, the plasma aspect ratio and elongation should not be modified from their default values (12.5 and 2.0, respectively). The whole machine can, however, be scaled in size. The recommended build-related iteration variables are as follows:

```
3 : rmajor
13 : tfcth
29 : bore
31 : gapomin
61 : gapds
```

All items external to the fusion power core (buildings, turbines, power conversion systems, etc.) remain unchanged.

3.5.2.2 Modelling of stellarator coils

The stellarator coils are assumed to be superconducting — no resistive coil calculations are performed. The overall calculations on the coils are largely unchanged, as the present superconducting routines still apply. However, the geometry of the coils is different from that for tokamak TF coils, so some modification has been necessary.

Firstly, the inboard legs do not form a continuous ring of material on the midplane — adjacent coil cases do not touch. This causes the calculation of the inboard coil conductor area to change, and also the shape of the winding pack cross-section. This is assumed to be rectangular for the stellarator coils, rather than the complicated two-step cross-section assumed for tokamaks (see Figure 3.6).

The radial thickness is set using `tfcth` as usual, but the toroidal thickness may also be set, using iteration variable no. 77 (`tftort`). This is constrained to be no larger than is geometrically possible using constraint equation no. 47 with iteration variable no. 76 (`frfptf`).

Secondly, the shape of the coils in the poloidal plane is not D-shaped, and varies from coil to coil. This has a number of implications, not least on the length (and hence total mass and cost) of the coil set. In order to model a set of realistically-shaped stellarator coils, geometry (θ, ϕ) data were taken from the Wendelstein VII-X coil set design [7] and these are fitted onto the surface of a helically-modulated torus whose radii scale with the machine's desired radial build. The total length of the coils is then calculated. For simplicity, the poloidal cross-section of the torus onto which the coils are mapped is assumed to be elliptical, with the major axis of the ellipse rotating with toroidal angle. This is only a rough estimate of the true cross-sectional shape; however the resulting coil set is a fair approximation to the true case (see Figure 3.7).

In order to use the existing routines for calculating the stresses in the coils, it is necessary to describe the poloidal variation of the coil shape using four circular arcs. For the purpose of these calculations the coil shape is assumed to be elliptical.

3.5.2.3 Other systems

Many of the fusion power core systems are assumed to have the same characteristics as for a tokamak device, including the blanket, divertor, cryogenic and vacuum systems. Of these, only the calculations for the divertor are likely to be inaccurate, as in a large stellarator device the divertor is expected to be helical rather than axisymmetric as is the case in tokamaks.

3.5.3 Limitations in the model

As may be already clear, there are a number of simplifications in the stellarator model, none of which should be too serious. These may be summarised as follows:

1. The plasma geometry calculations are not consistent with the stellarator situation in which the plasma cross-sectional shape varies with toroidal angle. This has knock-on effects on parameters such as the first wall area and the poloidal field B_p. Similarly, the density and temperature profiles must be thought of as averages over toroidal angle in this representation. Nevertheless, the use of mean thicknesses etc. is an acceptable approximation.

2. The cross-section of the coils in the poloidal plane is rather inaccurately modelled using ellipses. This has implications on the calculation of the coil masses and on the stresses on the coils. The calculated stresses are especially likely to be inaccurate as the stellarator coils are non-planar, so extra torque components will be present.

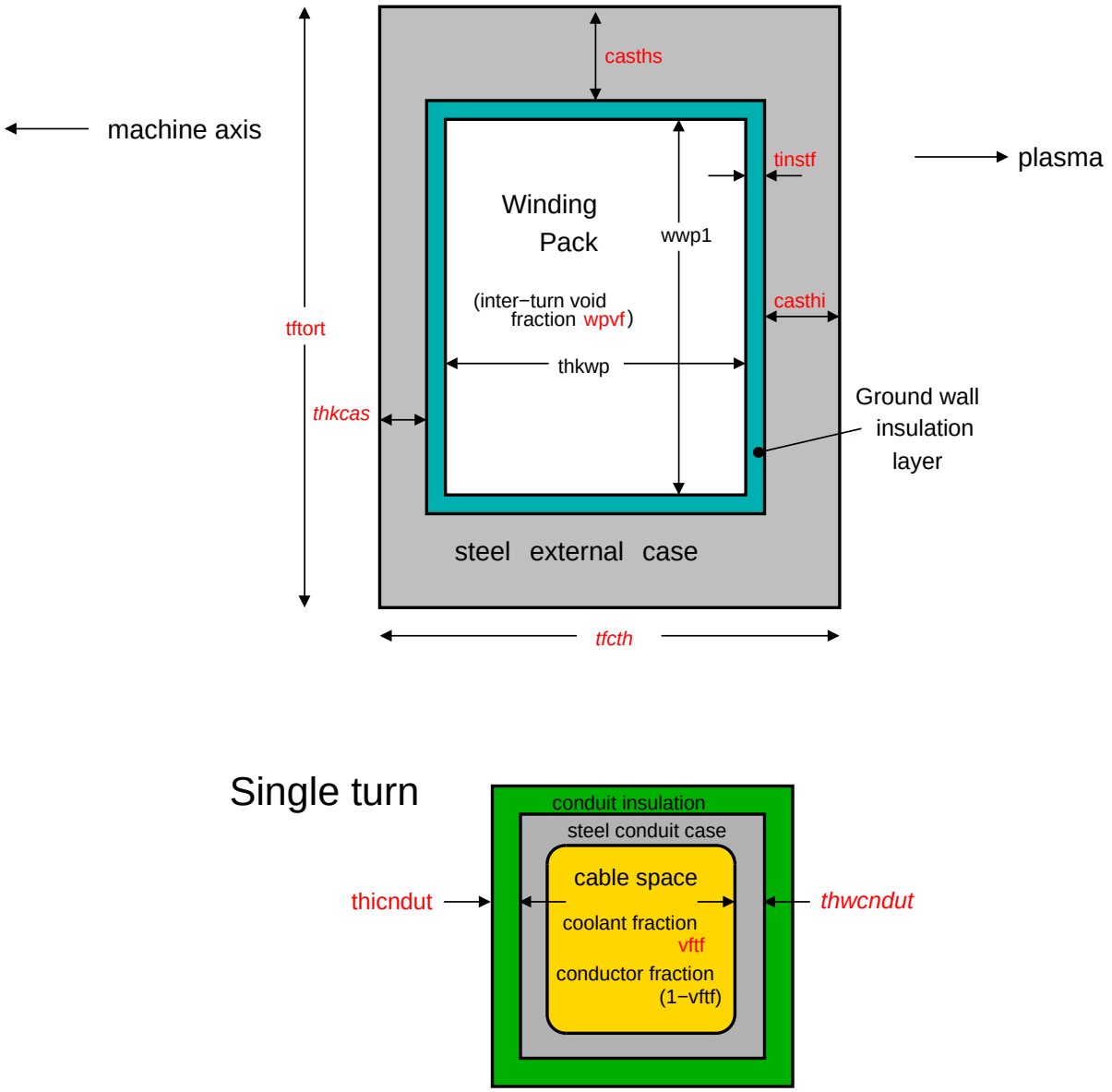

Figure 3.6: *Schematic diagram of the cross-section of the inboard leg of a superconducting stellarator coil, showing the CICC (Conductor In Cable Conduit) construction. The winding pack contains many turns of cable conduit. The cable space contains the superconducting filaments, and circulating liquid helium coolant. The variables shown in red may be changed by the user, and those in italics may be chosen as iteration variables.*

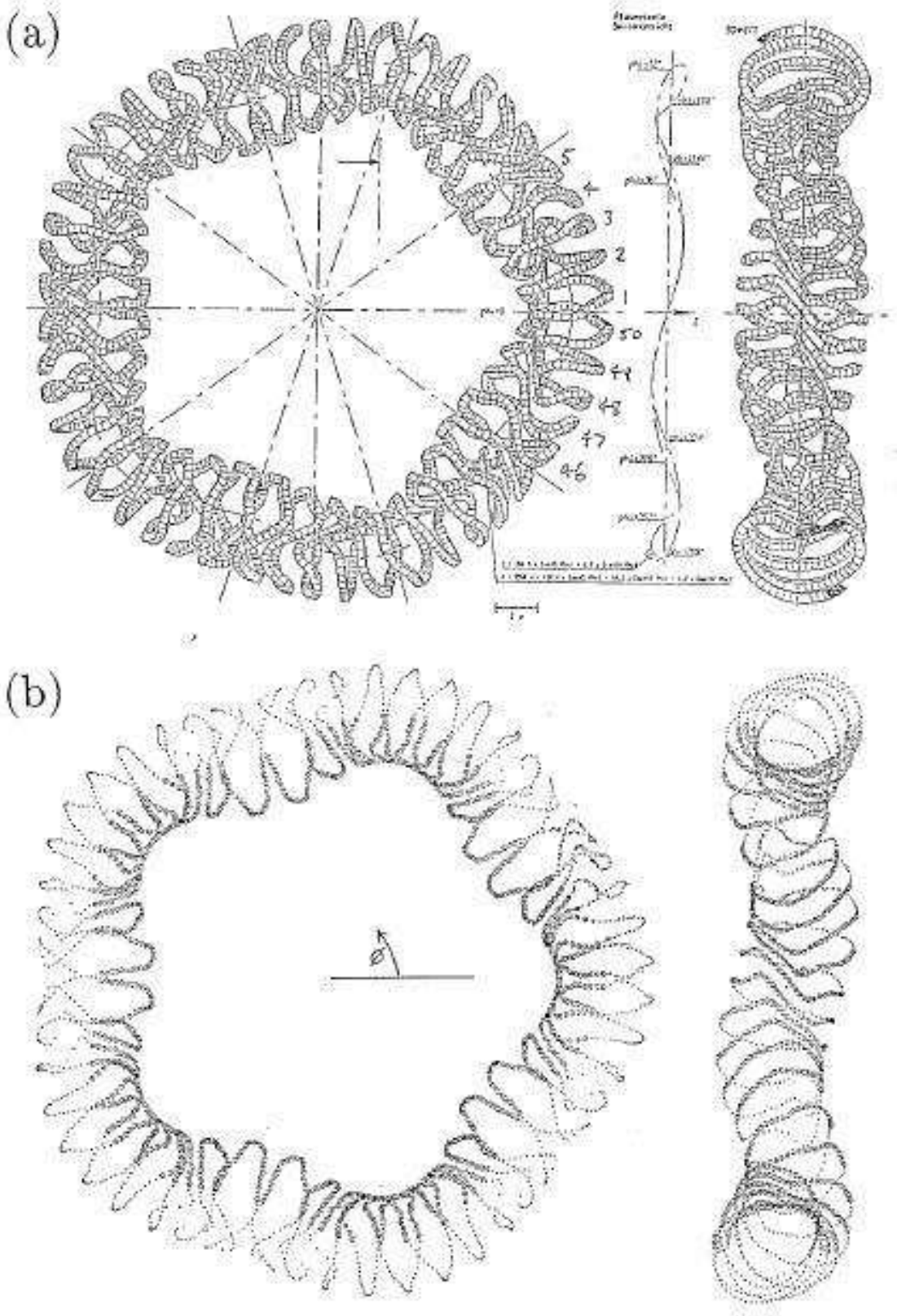

Figure 3.7: *Comparison between (a) the Wendelstein VII-X coil set and (b) the PROCESS stellarator coil set. The inboard sections of the PROCESS coils are plotted with circular symbols for clarity.*

3. The divertor model has not been modified from the axisymmetric tokamak case.

3.6 Reversed Field Pinch Model

In addition to the tokamak and stellarator magnetic confinement devices, the code has the ability to perform calculations based on the physics and engineering of a reversed field pinch (RFP) device. This third type of toroidal magnetic device is superficially similar in design to a tokamak (so therefore shares many of the same components), but the magnetic field configuration differs.

The model used in `PROCESS` is largely based on the TITAN fusion power plant [9, 10]. The following sections summarise its main features, where they differ from those for tokamaks.

To activate the RFP coding, it is necessary to create a file `device.dat`, containing the single character `2` in the first row, in the working directory (see Section 4.1). This has the effect of setting the internally-used switch `irfp = 1`. If the file is absent, or its first character is set to something other than `2`, the RFP model is not used, and `irfp` is set to `0`.

3.6.1 RFP physics

The plasma in RFPs is circular in cross-section and is axisymmetric.

3.6.1.1 Beta limit

The poloidal beta is limited to a maximum value given by input parameter `betpmx` (default = 0.19) using constraint equation 48 and iteration variable 79 (`fbetap`).

3.6.1.2 Density limit

No density limit is explicitly coded for RFPs, other than by simply constraining the upper bound of the electron density variable `dene` (iteration variable 6).

3.6.1.3 τ_E scaling law

One confinement time scaling law relevant to RFPs is present within `PROCESS`. The value of switch `isc` determines the scaling to be used in the plasma energy balance calculation.

$$\tau_E \text{ (TITAN RFP [9]: } \texttt{isc=25}) = 0.05\, a^2\, I_p(\text{MA})$$

3.6.1.4 F-Θ plot

Much of the RFP physics is derived from the characteristic F-Θ plot, where $F = B_\phi(a)/\langle B_\phi \rangle$, $\Theta = B_\theta(a)/\langle B_\phi \rangle$, and $\langle B_\phi \rangle$ is the average value of the toroidal field over the plasma cross-section. Given a value of the pinch parameter Θ, the corresponding value of the reversal parameter F may be read from the F-Θ plot, and from these the plasma current and the current in the TF coils may be obtained using

$$B_\theta(a) \quad = \quad \frac{\mu_0 I_p}{2\pi a}$$

$$B_\phi(a) \quad = \quad \frac{\mu_0 I_{TFC}}{2\pi R}$$

(the second of these assumes that $B_\phi(a)$ is approximately equal to the vacuum toroidal field at the plasma centre).

The pinch parameter Θ is set using iteration variable no. 78: `rfpth`. The corresponding value of the reversal parameter F (`rfpf`) is calculated using routine `FTHETA`. F is constrained to be negative using constraint equation no. 49 with f-value `frfpf` (iteration variable no. 80).

3.6.1.5 Current drive

The RFP oscillating field current drive option is turned on by setting `iefrf = 9`, with a fixed efficiency of 0.8 Amps per Watt of power injected into the plasma (the coding for this model is, therefore, trivial). The wall plug to injector efficiency is set using input parameter `etaof`, which has a default value of 0.5, and the unit cost is set using input parameter `ucof`. The default value for `ucof` is 3.3 $ per Watt of injected power.

The bootstrap fraction is assumed to be zero.

Plasma ignition, and additional plasma heating using auxiliary power, are treated as for tokamaks.

3.6.2 TF coils

The TF coils for the RFP option are derived from the TITAN-II coil set, which uses circular copper coils. The radial thickness is set using `tfcth` as usual, but the toroidal thickness may also be set, using iteration variable no. 77 (`tftort`). This is constrained to be no larger than is geometrically possible using constraint equation no. 47 with iteration variable no. 76 (`frfptf`).

3.6.3 OH coils

The TITAN-I OH coil locations, currents etc. are assumed. These comprise 14 copper coils, with currents swinging from positive to negative during the plasma start-up period, and then decaying back to zero. To account for different plasma geometries and currents, the OH coil locations relative to the plasma centre scale with the TF coil radius and with the plasma major radius, and the current per turn scales linearly with the plasma current.

3.6.4 EF coils

The Equilibrium Field coils are based on the TITAN-I EF coils, which are two superconducting (NbTi) coils that provide the correct vertical field at the plasma centre. The coil locations scale with the plasma major radius and TF coil radius along with the OH coils.

3.6.5 Divertor

The TITAN divertor concept uses three poloidal divertors, which bear little resemblance to the typical tokamak toroidal divertor systems. The code uses the TITAN divertor lifetime of one year to enable the divertor costs to be reasonable, although the divertor surface area (and therefore the cost per divertor) is likely to be inaccurate.

3.6.6 Code modifications

As with the stellarator model, the RFP model has been incorporated in such a way as to allow its simple removal again if required in the future. All new routines are confined to the dedicated source file `rfp.f90`. The tokamak-relevant consistency equations in `PROCESS` (see Section 2.4) are used without modification, to ensure that the coil currents and the fields they produce are consistent with the plasma parameters.

3.7 Inertial Fusion Energy Model

As well as magnetic confinement devices, `PROCESS` has the ability to model inertial fusion plants, in which a laser or ion beam is used to ignite a target pellet containing the fusion fuel.

To activate the inertial fusion energy (IFE) coding, it is necessary to create a file `device.dat`, containing the single character `3` in the first row, in the working directory (see Section 4.1). This has the effect of setting the internally-used switch `ife = 1`. If the file is absent, or its first character is set to something other than `3`, the IFE model is not used, and `ife` is set to `0`.

The IFE model [22] is controlled using two additional switches.

`ifetyp = 0 :` Generic device build

`ifetyp = 1 :` OSIRIS-type device build [23, 24, 25]

`ifetyp = 2 :` SOMBRERO-type device build [26, 27]

`ifetyp = 3 :` HYLIFE-II-type device build [28, 29, 30]

Switch `ifetyp` defines the type of device that is assumed; this varies widely between different conceptual designs. The generic type assumes a cylindrically symmetric device, while the other types are approximations to the builds of the given conceptual machines [31]. In general, the build from the centre of the device (at the target ignition location) is in the order: chamber, first wall, gap, blanket, gap, shield, gap, building wall. The user specifies the thicknesses of these regions, and also the materials that are present and in what proportions. [expand]

`ifedrv = -1 :` Driver efficiency and target gain are input as functions of driver energy

`ifedrv = 0 :` Driver efficiency and target gain are input

`ifedrv = 1 :` SOMBRERO laser drive efficiency and target gain assumed

`ifedrv = 2 :` OSIRIS heavy ion beam driver efficiency and target gain are assumed [32]

Switch `ifedrv` defines how the code calculates the driver efficiency and target gain — these are the primary outputs required from the physics part of the model. For the SOMBRERO and OSIRIS cases (`ifedrv = 1` and `ifedrv = 2`, respectively) the driver efficiency and gain are calculated from curves of these parameters as functions of the driver energy. For the `ifedrv = -1` case, the user provides the driver efficiency and target gain versus driver energy, via the two arrays `etave(1:10)` and `gainve(1:10)` respectively; the element number corresponds to the driver energy in MJ, and outside the range 1–10 MJ the curves are extrapolated linearly. Finally, for the `ifedrv = 0` case, the user inputs single values for the driver efficiency (`drveff`) and target gain (`tgain`).

Constraint equation no. 50 can be turned on to enable the ignition repetition rate to remain below a user-specified upper limit (`rrmax`); iteration variable no. 86 (`frrmax`) is the associated f-value (see Section 2.4). The other iteration variables relevant for the IFE model are nos. 81–85 (`edrive`, `drveff`, `tgain`, `chrad` and `pdrive` — see Table 2.3).

3.8 Safety and Environment Models

At present, the neutronics, activation and inventory calculations comprise the safety and environment models in the code.

The models comprising the safety and environmental calculations [18] within the code are all called from routine `FISPAC`. They are only performed once, at the end of each run, as they take a relatively long time to evaluate, and the results are only used for diagnostic purposes — no constraints are imposed at present to minimise doses, for instance.

N.B. These models are currently not available in the present version of the code.

3.8.1 Neutronics

The neutronics module predicts the neutron flux spectra in the inboard and outboard first wall and blanket components. The spectra are based on a simplified tokamak device that has a fixed ratio (= 1.5825) between the outboard blanket thickness and the inboard blanket thickness, and are scaled according to the actual thickness of the outboard blanket. This relatively limited, single-parameter approach is expected to be replaced by a more general method, which should allow a more accurate portrayal of the device being modelled by `PROCESS`.

3.8.2 Activation and inventory information

The code evaluates the consequences of exposing the power plant's materials to the calculated neutron fluxes, subject to the limitations imposed by the neutronics module. A library of neutron cross-sections and decay data is used to calculate the total activity, gamma-ray dose rate and decay heat output due to the materials' exposure to neutrons, both at the end of the plant's life and at a time 100 years later. These values are relevant to decommissioning and disposal studies, and additional parameters that can be obtained from the nuclide inventory will also be included as the need arises.

3.9 Cost Models

The cost accounting used by `PROCESS` combines methods [35] used in the TETRA code [1] and the Generomak [36] scheme. The costs are split into the standard accounting categories [37] generally used in the reporting of power plant costs. The best references for the algorithms used are [2], and source file `costs.f90` in the code itself.

The majority of the costed items have a unit cost associated with them. These values scale with (for example) power output, volume, component mass etc., and many are available to be changed via the input file. All costs and their algorithms correspond to 1990 dollars.

3.9.1 Cost options

3.9.1.1 N^{th} of a kind costs

The unit costs of the components of the fusion power core are relevant to "first-of-a-kind" items. That is to say, the items are assumed to be relatively expensive to build as they are effectively prototypes and specialised tools and machines have perhaps been made specially to create them. However, if a "production line" has been set up, and R & D progress has allowed more experience to be gained in constructing the power core components, the costs will be reduced as a result. Variable `fkind` may be used to multiply the raw unit costs of the fusion power core items (by a factor less than one) to simulate this cost reduction for an N^{th}-of-a-kind device. In other systems studies of fusion power plants [38], values for this multiplier have ranged from 0.5 to 0.8.

3.9.1.2 Level of safety assurance

Many of the unit costs have four possible choices, relating to the level of safety assurance [39] flag `lsa`. A value `lsa = 1` corresponds to a plant with a full safety credit (i.e. is truly passively safe). Levels `2` and `3` lie between the two extremes, and level `4` corresponds to a present day fission reactor, with no safety credit.

3.9.1.3 Replaceable components

The first wall, blanket, divertor, centrepost (if present) and current drive system have relatively short lifetimes because of their hostile environment, after which they must be replaced. Because of this frequent renewal they can be regarded as though they are "fuel" items, and can be costed accordingly. Switch `ifueltyp` is used to control whether this option is used in the code. If `ifueltyp = 1`, the costs of the first wall, blanket, divertor and a fraction `fcdfuel` of the cost of the current drive system are treated as fuel costs. If `ifueltyp = 0`, these are treated as capital costs.

3.9.1.4 Cost of electricity calculations

Switch `ireactor` determines the type of cost of electricity calculation that is performed. If `ireactor = 0`, no cost of electricity calculation is performed. If `ireactor = 1`, then the cost of electricity is evaluated, with the value quoted in units of m$/kWh.

3.9.1.5 Net electric power calculation

Related to the cost of electricity is the net electric power calculation performed in routine `POWER`. It is possible that the net electric power can become negative due to a high recirculating power. Switch `ipnet` determines whether the net electric power is scaled to always remain positive (`ipnet = 0`), or whether it is allowed to become negative (`ipnet = 1`), in which case no cost of electricity calculation is performed.

3.10 Other Switches and Models

3.10.1 Output control

Since the user may only be interested in a small proportion of the code's output, a set of switches exists that controls whether a given section of the output file is produced. Table 3.6 indicates how these switches affect the output.

switch	relevant output section
sect01	power plant costs
sect02	detailed costings
sect03	plasma
sect04	current drive system
sect05	divertor
sect06	machine build
sect07	TF coils
sect08	PF coils
sect09	volt second consumption
sect10	support structure
sect11	PF coil inductances
sect12	shield / blanket
sect13	power conversion
sect14	heat transport
sect15	vacuum system
sect16	plant buildings
sect17	AC power
sect18	neutral beams
sect19	electron cyclotron heating
sect20	Lower Hybrid heating
sect21	times

Table 3.6: *Summary of the switches in PROCESS that control the format of the output file. If a switch has a value 0, the relevant output section does not appear in the output file. If its value is 1, the output section is included in the output file — this is the default setting for all the switches.*

3.10.2 Code parameters affecting other models

This chapter has summarised the methods by which several of the models in the code can be activated. There are many others present, however, and it is suggested that the user refers to the variable descriptor file, vardes.html. As stated earlier, this contains details of all the parameters within the code that can be changed by the user, in order to customise the machine modelled by PROCESS.

Chapter 4

Execution of the Code

The intention of this chapter is to provide a comprehensive prescription for setting up and performing runs with the code. Firstly, the input file's structure and format is described. The user is then taken through the process of setting up the code to model a new machine, and finally an attempt is made to indicate and solve the problems that the user will face whilst trying to achieve a feasible solution.

4.1 The Input File

The input file `IN.DAT` is used to change the values of the physics, engineering and other code parameters from their default values, and to set up the numerics (constraint equations, iteration variables etc.) required to define the problem to be solved.

4.1.1 Tokamak, stellarator, RFP or IFE?

In addition to the main input file `IN.DAT`, a second input file `device.dat` is used to signal to the code whether a tokamak, stellarator, reversed field pinch or inertial fusion enery plant is to be modelled. (If the file does not exist in the working directory, the standard tokamak model is used.)

File `device.dat` should contain a single character in the first line, which is interpreted as follows:

`0` : use tokamak model
`1` : use stellarator model
`2` : use reversed field pinch model
`3` : use inertial fusion energy model

4.1.2 File structure

The input file comprises a number of lines reminiscent of the Fortran `NAMELIST` format used by some programs to read in data. Except for comment lines that start with a `*` character, each line is of the form

```
variable = value
```

where `variable` is the name of one of the input parameters or iteration variables listed in the variable descriptor file, and `value` is the (usually numerical) initial value required for that variable. (Arrays,

as opposed to scalar quantities, are treated differently — see below.) Variables can be specified in any order in the input file.

The routines that read in and parse the data from the input file are in source file `input.f90`. They allow a great deal of error-trapping to be carried out at the input stage. All input data are screened for non-sensible values directly; this is a useful feature of the code, since without such intervention modern computers are notorious at not terminating programs when an arithmetically impossible or undefined operation ("`NaN`" error) is encountered.

4.1.3 Format rules

The following rules must be obeyed when writing an input file:

1. Each variable must be on a separate line.
2. Variable names can be upper case, lower case, or a mixture of both.
3. Spaces may not appear within a variable name or data value.
4. Other spaces within a line, and trailing spaces, are ignored.
5. Commas are not necessary between variables.
6. Data can extend over more than one line.
7. One-dimensional arrays can be explicitly subscripted, or unscripted, in which case the following element order is assumed: `A(1), A(2), A(3),...`
8. At present, multiple dimension arrays can only be handled without reference to explicit subscripts, in which case the following element order is assumed: `B(1,1), B(2,1), B(3,1),...` The use of the input file to specify multiple dimension array elements is prone to error.
9. Unscripted array elements must be separated by commas.
10. Blank lines are allowed anywhere in the input file.
11. Lines starting with a `*` are assumed to be comments.
12. Comment lines starting with five or more asterisks (i.e. `*****`) are reproduced verbatim in the output file. These should be used copiously to give a great deal of information about the run being performed, and should be updated before every single run of the code, as it is very easy to lose track of what is being attempted.

It is useful to divide the input file into sections, using suitable comment lines, to help the user keep related variables together.

The following is a valid fragment of an input file (the vertical lines are simply to help show the column alignment):

```
* This line is a comment that will not appear in the output
***** This line is a comment that will appear in the output
boundl(1) = 2.5,
BOUNDU(10) = 3.,
BOUNDU(45) = 1,
* Another comment...  Note that real values can be entered as if
* they were integers, but NOT vice versa.
epsfcn = 10.e-4,
Ftol = 1.D-4,
* The next line sets the first five elements of array icc:
ICC = 2, 10, 11, 24, 31
* The next line sets the first ten elements of array ixc:
ixc = 10, 12, 3, 36, 48,
       1, 2, 6, 13, 16,
IOPTIMZ = 1,
maxcal = 200
NEQNS = 5,
NVAR = 10,
```

The following are *invalid* entries in the input file (Q: Why?!):

```
boundl(1,1) = 2.5,
BOUNDU(N) = 3.,
A line of 'random' characters like this will clearly wreak havoc
eps fcn = 10.e-4, ftol = 1.D-4
epsvmc = 1.0 e-4
ICC = 2 10 11 24 31
IOPTIMZ = 1.0,
```

4.2 The Output File

The output from the code is sent to file `OUT.DAT` in the working directory.

4.3 Running the Code

This section will attempt to guide the user through the actual running of the code in its various modes. In most cases only minor changes to the input file are necessary to change the code's mode of operation — usually the physics and engineering variables, etc. remain unchanged, with the major differences occurring in the numerical input only.

4.3.1 Non-optimisation mode

Non-optimisation mode is used to perform benchmark comparisons, whereby the machine size, output power etc. are known and one only wishes to find the calculated stresses, beta values and fusion powers, for example. When starting to model a new machine, `PROCESS` should always be run first in non-optimisation mode, before any attempt is made to optimise the machine's parameters.

The first thing to do is to add to the input file all the known details about the machine to be modelled. This may include some or all of the following:

- machine build
- plasma aspect ratio
- PF coil locations
- type of current drive to be used
- net electric power
- various physics parameters, e.g.
  - toroidal field on axis
  - electron density
  - electron temperature
  - elongation
  - triangularity
  - Troyon g coefficient
  - edge safety factor

In addition, some of the switch values summarised in Chapter 3 may have to be altered from their default values.

Next, the relevant numerics information must be entered. Switch `ioptimz` must be set to `-1` for non-optimisation mode. Then the user must decide which constraint equations and iteration variables to activate — this choice is dictated partly by the information required by the user, and partly by the machine being modelled itself.

As stated earlier, all the relevant consistency equations must be activated, together with the corresponding iteration variables. A number of limit equations can also be activated, to investigate how the calculated values compare with the physics or engineering limits. The following is part of an example non-optimisation input file:

```
IOPTIMZ = -1
NEQNS = 8
NVAR = 8
ICC =  1,  2, 10, 11,  7, 16,  5, 24,
IXC =  5, 10, 12, 29,  7,      9, 36, 4,

FPNETEL = 1.0
PNETELIN = 1200.0
⋮
```

Consistency equations 1, 2, 10, 11 and 7 are activated, together with limit equations 16, 5 and 24 (refer to Table 2.1). This example assumes that neutral beam current drive is present (equation 7 with variable 7), and that the net electric power is to be fixed at 1200 MW. Note the optional (but beneficial) practice of vertically aligning corresponding equations and variables — constraint equation 16 has no corresponding iteration variable (which would normally be no. 25, `fpnetel`), as we want the net electric power to be fixed at the value given by `pnetelin`. Since in non-optimisation mode, the number of variables must be equal to the number of equations, we have scope to add a "free" iteration variable (refer to Tables 2.2 and 2.3), in this case no. 4 — electron temperature, to help raise the fusion power sufficiently to obtain the required net electric power. Finally, note the

use of the density and Troyon beta limit equations (`5` and `24`, respectively); the final values of the corresponding f-values will indicate if the limits are exceeded and by how much.

On running `PROCESS` and (hopefully) achieving a feasible result, examination of the output may well show up discrepancies between some of the parameter values produced and their known values (if available). Remember that, of all the variables shown in the variable descriptor file with a default value, only those declared as *active iteration variables* can change from their initial values, whether they are set in the input file or in the relevant module. However some of the calculated parameters may be wrong, the most common of which are as follows:

- Plasma current; this can be adjusted using the edge safety factor `q`: $I_p \propto 1/q$
- Fusion power; this scales roughly with the density profile factor `alphan`
- Build parameters; it may be necessary to change non-critical thicknesses to achieve the correct machine build.

It may still be difficult, if not impossible, to reconcile the fusion power and the net electric power with the required values. This may well be due to the power conversion efficiency values being used — refer to Figure 3.5.

With luck, a few iterations of this process will produce an adequate benchmark case. A typical input file for use with `PROCESS` in non-optimisation mode is contained in Appendix A.

4.3.2 Optimisation mode

Running `PROCESS` in optimisation mode requires few changes to be made to the input file from the non-optimisation case. The main differences between optimisation mode and non-optimisation mode are:

1. Optimisation mode applies lower and upper bounds to all active iteration variables.
2. There is no upper limit to the number of active iteration variables in optimisation mode.
3. A figure of merit must be specified in optimisation mode.
4. Scans can be performed in optimisation mode.

Switch `ioptimz` must be set to `0` or `1` for optimisation mode. If `ioptimz = 0`, a non-optimisation pass is performed first, to provide a (hopefully) feasible set of initial conditions; if `ioptimz = 1`, this is skipped and the code runs in optimisation mode from the start. It is usually recommended to use `ioptimiz = 0` for optimisation runs.

As before, the user must decide which constraint equations and iteration variables to activate. Again, the choice depends largely on the information required by the user and the extent of the freedom that the code may have with the machine's parameters.

The following is part of an example optimisation input file:

```
IOPTIMZ = 1
NEQNS = 16
NVAR = 19
ICC =  1,  2, 10, 11,  7, 16,  5, 24, 14,  8, 31, 32, 33, 34, 35, 36,
```

```
IXC =  5, 10, 12, 29,  7,      9, 36, 19, 14, 48, 49, 50, 51, 53, 54,
4, 6, 1, 18,
BOUNDL(1) = 2.5
BOUNDU(10) = 2.0
MINMAX = 6

FPNETEL = 1.0
PNETELIN = 1200.0
WALALW = 4.4

ISWEEP = 3
NSWEEP = 11
SWEEP = 3.5, 3.7, 3.9
:
```

The figure of merit in this example is the (minimum) cost of electricity (`minmax = 6`). Note that additional limit equations are now active, along with a second consistency equation related to the neutral beam current drive — the number of decay lengths to the plasma centre is constrained to be equal to the input value (`tbeamin`, which is not shown here). Furthermore, there are now more iteration variables than constraint equations, to aid the minimisation process. Finally, note that a three-point scan in the Troyon g coefficient `dnbeta` — scanning variable 11, is to be performed.

A useful practice in optimisation mode is to perform "stationary" scans, whereby the same value is given to the scanning variable on successive iterations. This provides a check as to how well converged the solution has become. If scans of a given variable are to be made over a large range of values, it is often a good idea to start the scan in the middle of the desired range, and to split the scan in two — one going downwards from the initial value, and the other upwards. This ensures that the whole range of the scan produces well-converged machines (assuming a "good" initial point), without sharp changes in gradient in the parameter values.

It should be remembered that the value of the scan variable is set in the array `sweep`, and this overrules any value set for the variable elsewhere in the input file. For instance, in the example above, the values of `dnbeta` set in the `sweep` array would overrule any value for `dnbeta` set elsewhere in the file.

The output from an optimisation run contains an indication as to which iteration variables lie at their limiting values. On the whole there is a greater chance of unfeasible solutions being found whilst in optimisation mode, and Section 4.4 will hopefully be of some use in this situation. A typical input file for use with `PROCESS` in optimisation mode is contained in Appendix B.

4.4 Problem Solving

Experience has shown that the first few attempts at running `PROCESS` with a new input file tends to produce unfeasible results — that is, the code will not find a consistent set of machine parameters. The highly non-linear nature of the numerics of `PROCESS` is the reason for this difficulty, and it often requires a great deal of painstaking adjustment of the input file to overcome.

4.4.1 General problems

A code of the size and complexity of `PROCESS` contains myriads of equations and variables. Virtually everything depends indirectly on everything else because of the nature of the code structure, so perhaps it is not surprising that it is often difficult to achieve a successful outcome.

Naturally, problems will occur if some of the parameters become unphysical. For example, if the aspect ratio becomes less than or equal to one, then we must expect problems to appear. For this reason, the default bounds on the iteration variables and the allowed ranges of all the input variables have been selected with great care.

The code contains a large (though probably not exhaustive) number of error traps to try and prevent problems from propagating. These include tests for unphysical values, and checks to prevent divisions by zero, and non-sensible arguments for logarithms and square roots, etc. However, occasionally arithmetic ("`NaN`") errors still occur, although their incidence is low. They now usually only occur due to unfeasibility problems (see later).

The error messages produced by the code attempt to provide diagnostic information, telling the user where the problem occurs, and also suggest a possible solution. These messages are out of necessity brief, and so cannot promise to lead to a more successful outcome.

4.4.2 Optimisation problems

On reflection it is perhaps surprising that `PROCESS` ever does manage to find the global minimum figure of merit value, since if there are `nvar` iteration variables active the search is over `nvar`-dimensional parameter space, in which there may be many shallow minima of approximately equal depth. Remember that `nvar` is usually of the order of twenty.

The machine found by `PROCESS` may not, therefore, be the absolutely optimal device. It is quite easy to have two or more solutions, with results only a few per cent different, but a long way apart in parameter space. The technique of "stationary" scans described in Section 4.3.2 above can often help in this situation, which is why this method is recommended at all times.

Scans should be started in the middle of a range of values, to try to keep the scan within the same family of machines. The optimum machine found may otherwise suddenly jump to a new region of parameter space, causing the output variables to seem to vary unpredictably with the scanning variable.

It should be noted that in general the machine produced by `PROCESS` will always sit against one or more operation limits. If, during a scan, the limit being leant upon changes (i.e. if the machine jumps from leaning on the beta limit to leaning on the density limit) the output parameters may well become discontinuous in gradient, and trends may suddenly change direction.

4.4.3 Unfeasible results

In the numerics section of the output file, the code indicates whether the run produced a feasible or unfeasible result.

The former implies a successful outcome, although it is always worth checking that the estimate of the constraints (`sqsumsq`) is small ($\sim 10^{-3}$ or less); the code will issue a warning if the run seems feasible but the value of `sqsumsq` exceeds 10^{-2}. If this occurs, reducing the value of the HYBRID tolerance `ftol` or VMCON tolerance `epsvmc` (as appropriate) should indicate whether the result is valid or not; the output can usually be trusted if (1) the constraint residues[1] fall as the tolerance is reduced to about 10^{-8}, and (2) the code indicates that a feasible solution is still found.

An unfeasible result occurs if `PROCESS` cannot find a set of values for the iteration variables which satisfies all the given constraints. In this case, the values of the constraint residues shown in the output

[1]The constraint residues are the final values of c_i in the constraint equations — see Section 2.4. The value `sqsumsq` is the square root of the sum of the squares of these residuals.

give some indication of which constraint equations are not being satisfied — those with the highest residues should be examined further. In optimisation mode, the code also indicates which iteration variables lie at the edge of their allowed range.

Unfeasible runs are caused either by ill-defining the problem to be solved, or by starting the problem in an unfavourable region of parameter space. The latter can be checked simply by changing the initial values of the *active* iteration variables in the input file, but the former requires some extra work. This situation arises if there are insufficient iteration variables for the given constraint equations. It is important to choose the right number of *useful* iteration variables for the problem to be solved — it is possible to activate too many iteration variables as well as too few, some of which may be redundant.

Both optimisation and non-optimisation runs can fail with an error message suggesting that the iteration process is not making good progress. This is likely to be due to the code finding itself unable to escape a region of the parameter space where the minimum in the residuals is significantly above zero. In this situation, there is either no solution possible (the residuals can therefore never approach zero), or the topology of the local minimum makes it difficult for the code to escape to the global minimum. Again, a helpful technique is to either change the list of iteration variables in use, or to simply modify their initial values to try to help the code avoid such regions.

A technique that occasionally removes problems due to unfeasible results, particularly if an error code `ifail = 3` is encountered during an optimisation run, is to adjust slightly one of the limits imposed on the iteration variables, even if the limit in question has not been reached. This subtly alters the gradients computed by the code during the iteration process, and may tip the balance so that the code decides that the device produced is feasible after all. For instance, a certain component's temperature might be 400 K, and its maximum allowable temperature is 1000 K. Adjusting this limit to 900 K (which will make no difference to the *actual* temperature) may be enough to persuade the code that it has found a feasible solution.

Similarly, the order in which the constraint equations and iteration variables are stored in the `icc` and `ixc` arrays can make the difference between a feasible and unfeasible result. This seemingly illogical behaviour is, sadly, typical of the way in which the code works.

Note added 17/01/2013: It has been found that, during optimisation runs, the iteration variable indicated by the final specified element in array `ixc` (i.e. `ixc(nvar)`) does not finish the run with consistent values in the output file (its value is roughly 0.1% less than the correct value). The reason is partly understood, but difficult to correct in the source code. A quick fix is to increment `nvar` by one, and add another iteration variable to the `end` of `ixc` which will have no effect on the convergence; for instance, use an f-value for a constraint equation which has not been turned on. Then it will be this redundant variable which will have an inconsistency, but this is of no consequence.

Unfeasible cases often produce unrealistic machines, so one should not believe the output values from these runs. Unfortunately, the stationary scan method sometimes, though not always, fails to help these cases, since it will tend to keep starting the run at the same point. Ill-defined problems sometimes produce arithmetic errors, for obscure reasons.

Though a great deal of work has been performed on the code to improve its standard, there can be no guarantee that `PROCESS` is entirely bug-free, simply because of its large size. Rarely, then, it may be that an unfeasible result indicates that the code has encountered a programming error, although its precise location will be almost impossible to find by simply examining the output file.

It may be the case that the act of satisfying all the required constraints is impossible. No machine can exist if the allowed operating regime is too restrictive, or if two constraint equations require conflicting parameter spaces. In this case some relaxation of the requirements is needed for the code to produce a successful machine design.

4.4.4 Hints

The above sections should indicate that it is the complex interplay between the constraint equations and the iteration variables that determines whether the code will be successful at producing a useful result. It can be a somewhat laborious process to arrive at a working case, and (unfortunately, perhaps) experience is often of great value in this situation.

It should be remembered that sufficient iteration variables should be used to solve each constraint equation. For instance, a particular limit equation may be $A \leq B$, i.e. $A = fB$, where the f-value f must lie between zero and one for the relation to be satisfied. However, if none of the iteration variables have any effect on the values of A and B, and A happens to be *greater* than B, then `PROCESS` will clearly not be able to solve the constraint.

The lower and upper bounds of the iteration variables are all available to be changed in the input file. Constraints can be relaxed in a controlled manner by moving these bounds, although in some cases care should be taken to ensure that unphysical values cannot occur. The code indicates which iteration variables lie at the edge of their range.

It is suggested that constraint equations should be added one at a time, with sufficient new iteration variables activated at each step. If the situation becomes unfeasible it can be helpful to reset the initial iteration variable values to those shown in the output from a previous feasible case, and rerun the code.

Finally, it should be borne in mind that the machine that is envisaged may not be a valid solution to the constraints being imposed, no matter how many degrees of freedom (i.e. iteration variables) are available. In this case, and many others, the user has to relax the constraints slowly until a feasible result is found.

Chapter 5

Inclusion of Additional Variables and Equations

It is often useful to add extra features to the code in order to model new situations. This chapter provides instructions on how to add various numerics related items to `PROCESS`.

Please remember to modify the relevant Table(s) in this User Guide if changes are made to the source code!

5.1 Input Parameters

Input parameters (see Section 2.3) are added to the code in the following way:

1. Choose the most relevant module (usually one of those in source file `global_variables.f90`). Keeping everything in alphabetical order, add a declaration statement for the new variable, specifying a "sensible" default value, and a correctly formatted comment line to describe the variable (copying the examples already present).
2. Ensure that all the modules that use the new variable reference the relevant module via the Fortran `use` statement.
3. Add the parameter to routine `PARSE_INPUT_FILE` in source file `input.f90` in a suitable place — keep to alphabetical order. The existing examples provide guidance on how to do this. Note that real (i.e. double precision) and integer variables are treated differently, as are scalar quantities and arrays.

5.2 Iteration Variables

New iteration variables (see Section 2.5) are added in the same way as input parameters, with the following additions:

1. Increment the parameter `ipnvars` in module `numerics` in source file `numerics.f90` to accommodate the new iteration variable.
2. Add an additional line to the initialisation of the array `ixc` in module `numerics` in source file `numerics.f90`.

3. Assign sensible values for the variable's bounds to the relevant elements in arrays `boundl` and `boundu` in module `numerics` in source file `numerics.f90`.

4. Assign the relevant element of character array `lablxc` to the name of the variable, in module `numerics` in source file `numerics.f90`.

5. Add the variable to routines `LOADXC` and `CONVXC` in source file `iteration_variables.f90`, mimicking the way that the existing iteration variables are coded.

Don't forget to add suitable correctly-formatted comment lines to `numerics.f90` to document the above changes.

If an existing input parameter is now required to be an iteration variable, then simply carry out the tasks mentioned here.

It should be noted that iteration variables must not be reset elsewhere in the code. That is, they may only be assigned new values when originally initialised (in the relevant module, or in the input file if required), and in routine `CONVXC` where the iteration process itself is performed. Otherwise, the numerical procedure cannot adjust the value as it requires, and the program will fail.

5.3 Other Global Variables

This type of variable embraces all those present in the modules in `global_variables.f90` (and some others elsewhere) which do not need to be given initial values or to be input, as they are calculated within the code. These should be added to the code in the following way:

1. Choose the most relevant module (usually one of those in source file `global_variables.f90`). Keeping everything in alphabetical order, add a declaration statement for the new variable, specifying the initial value `0.0D0`, and a correctly formatted comment line to describe the variable (copying the examples already present).

2. Ensure that all the modules that use the new variable reference the relevant module via the Fortran `use` statement.

5.4 Constraint Equations

Constraint equations (see Section 2.4) are added to `PROCESS` in the following way:

1. Increment the parameter `ipeqns` in module `numerics` in source file `numerics.f90` to accommodate the new constraint.

2. Add an additional line to the initialisation of the array `icc` in module `numerics` in source file `numerics.f90`.

3. Assign a description of the new constraint to the relevant element of array `lablcc`, in module `numerics` in source file `numerics.f90`.

4. Add the constraint equation to routine `CONSTRAINTS` in source file `constraint_equations.f90`, ensuring that all the variables used in the formula are contained in the modules specified via `use` statements present at the start of this routine. Use a similar formulation to that used for the existing constraint equations, remembering that the code will try to force `cc(i)` to be zero.

Don't forget to add suitable correctly-formatted comment lines to `numerics.f90` to document the above changes.

Remember that if a limit equation is being added, a new f-value iteration variable may also need to be added to the code.

5.5 Figures of Merit

New figures of merit (see Section 2.6) are added to `PROCESS` in the following way:

1. Increment the parameter `ipnfoms` in module `numerics` in source file `numerics.f90` to accommodate the new figure of merit.
2. Assign a description of the new figure of merit to the relevant element of array `lablmm` in module `numerics` in source file `numerics.f90`.
3. Add the new figure of merit equation to routine `FUNFOM` in source file `evaluators.f90`, following the method used in the existing examples. The value of `fc` should be of order unity, so select a reasonable scaling factor if necessary. Ensure that all the variables used in the new equation are contained in the modules specified via `use` statements present at the start of this file.

Don't forget to add suitable correctly-formatted comment lines to `numerics.f90` to document the above changes.

5.6 Scanning Variables

Scanning variables (see Section 2.7) are added to `PROCESS` in the following way:

1. Increment the parameter `ipnscnv` in module `scan_module` in source file `scan.f90` to accommodate the new scanning variable.
2. Add a short description of the new scanning variable to the `nsweep` entry in source file `scan.f90`.
3. Add a new assignment to the relevant part of routine `SCAN` in source file `scan.f90`, following the examples already present, including the inclusion of a short description of the new scanning variable in variable `xlabel`.
4. Ensure that the scanning variable used in the assignment is contained in one of the modules specified via `use` statements present at the start of this routine.

Chapter 6

Code Management Utilities

6.1 Makefile

6.2 Automatic Documentation

Chapter 7

Acknowledgements & Bibliography

The author would like to thank the following people for many useful and revealing discussions during his work on PROCESS:

— John D. Galambos, Paul C. Shipe and Y-K. Martin Peng of Oak Ridge National Laboratory, USA,

— Roger Hancox (now retired), Neill Taylor, Robin Forrest and Ian Cook of Fusion Physics Department, UKAEA Fusion,

— John Hicks of Engineering Department, UKAEA Fusion,

— Chris Gardner, formerly of Microwave and Interpretation Department, UKAEA Fusion,

— Tim Hender of Microwave and Interpretation Department, UKAEA Fusion, and all the co-authors of reference [15].

This work was funded by the UK Department of Trade and Industry, Euratom, and by internal research funds of UKAEA.

Bibliography

[1] R. L. Reid et al., *"ETR/ITER Systems Code"*, Oak Ridge Report ORNL/FEDC-87/7 (1988)

[2] J. D. Galambos, *"STAR Code : Spherical Tokamak Analysis and Reactor Code"*, Unpublished internal Oak Ridge document. A copy exists in the `PROCESS` Project Work File [3].

[3] N. P. Taylor (holder) and P. J. Knight, *"`PROCESS` Reactor Systems Code"*, AEA Fusion Project Work File, F/RS/CIRE5523/PWF (1992)

[4] L. Bottura, *"$J_c(B, T, \epsilon)$ Parameterizations for the ITER Nb_3Sn Production"*, ITER Document 2MMF7J (2008), `https://user.iter.org/?uid=2MMF7J&action=get_document`

[5] Y-K. M. Peng and J. B. Hicks, *"Engineering Feasibility of Tight Aspect Ratio Tokamak (Spherical Torus) Reactors"*, AEA Fusion Report AEA FUS 64 (1990)

[6] J. F. Lyon, K. Gulec, R. L. Miller and L. El-Guebaly: *"Status of the U.S. Stellarator Reactor Study"*, report communicated privately to T. Hender

[7] Wendelstein Project Group, *"Wendelstein VII-X: Application for Preferential Support"*, IPP-Euratom Association, August 1990

[8] G. Grieger et al., *Fusion Technology*, **21** (1992) 1767

[9] *"The TITAN Reversed Field Pinch Fusion Reactor Study, Scoping Phase Report"* , UCLA Report UCLA-PPG-1100, January 1987

[10] *"The TITAN Reversed Field Pinch Fusion Reactor Study, Final Report"*, UCLA Report UCLA-PPG-1200, 1990

[11] J. J. More, B. S. Garbow and E. Hillstrom, *"User Guide for MINPAC-1"*, Argonne National Laboratory Report ANL-80-74 (1980)

[12] M. J. D. Powell, *"A Hybrid Method for Non-linear Algebraic Equations"*, Numerical Methods for Non-linear Algebraic Equations, ed. P. Rabinowitz, Prentice-Hall

[13] R. L. Crane, K. E. Hillstrom and M. Minkoff, *"Solution of the General Nonlinear Programming Problem with Subroutine VMCON"*, Argonne National Laboratory Report ANL-80-64 (1980)

[14] N. A. Uckan and ITER Physics Group, *"ITER Physics Design Guidelines: 1989"*, ITER Documentation Series, No. 10, IAEA/ITER/DS/10 (1990)

[15] T. C. Hender, M. K. Bevir, M. Cox, R. J. Hastie, P. J. Knight, C. N. Lashmore-Davies, B. Lloyd, G. P. Maddison, A. W. Morris, M. R. O'Brien, M. F. Turner and H. R. Wilson, *"Physics Assessment for the European Reactor Study"*, AEA Fusion Report AEA FUS 172 (1992)

[16] W. M. Nevins et al., *"Summary Report: ITER Specialists' Meeting on Heating and Current Drive"*, ITER-TN-PH-8-4, 13–17 June 1988, Garching, FRG

[17] H. R. Wilson, Nuclear Fusion **32** (1992) 257

[18] N. P. Taylor, R. A. Forrest, P. J. Knight and L. J. Baker, *"Safety and Environmental Modelling in the PROCESS Code"*, Strategic Studies Note 94/14 (1994)

[19] N. A. Uckan et al., Fusion Technology **13** (1988) 411

[20] J. Nührenberg et al., *Plasma Physics and Controlled Fusion*, **35** (1993) B115

[21] S. Sudo, Y. Takeiri, H. Zushi et al., *Nuclear Fusion*, **30** (1990) 11

[22] P. J. Knight, *"PROCESS 3009: Incorporation of Inertial Fusion Energy Model"*, Work File Note F/MI/PJK/PROCESS/CODE/032

[23] Bourque et al., *"Overview of the OSIRIS IFE Reactor Conceptual Design"*, Fusion Technology **21** (1992) 1465

[24] Meier and Bieri, *"Economic Modeling and Parametric Studies for OSIRIS — a HIB-Driven IFE Power Plant"*, Fusion Technology **21** (1992) 1547

[25] Ghose et al., *"BOP Designs for OSIRIS and SOMBRERO IFE Reactor Plants"*, Fusion Technology **21** (1992) 1501

[26] Sviatoslavsky et al., *"A KrF Laser Driven Inertial Fusion Reactor SOMBRERO"*, Fusion Technology **21** (1992) 1470

[27] Meier and Bieri, *"Economic Modeling and Parametric Studies for SOMBRERO — a Laser-Driven IFE Power Plant"*, Fusion Technology **21** (1992) 1552

[28] Moir et al., *"HYLIFE-II: A Molten-Salt Inertial Fusion Energy Power Plant Design — Final Report"*, Fusion Technology **25** (1994) 5

[29] Moir, *"HYLIFE-II Inertial Fusion Energy Power Plant Design"*, Fusion Technology **21** (1992) 1475

[30] Hoffman and Lee, *"Performance and Cost of the HYLIFE-II Balance of Plant"*, Fusion Technology **21** (1992) 1475

[31] P. J. Knight, *"PROCESS IFE Build Details"*, F/MI/PJK/LOGBOOK12, pp. 52, 53, 56, 57

[32] Bieri and Meier, *"Heavy-Ion Driver Parametric Studies and Choice of a Base 5 MJ Driver Design"*, Fusion Technology **21** (1992) 1557

[33] P. J. Karditsas, *"`PROCESS` Code Development: simple blanket model — neutron heat input"*, Work File Note F/RS/CIRE5523/PWF/0142, November 1992

[34] *"Pulsed Fusion Reactor Study"*, AEA Fusion Report AEA FUS 205 (1992)

[35] R. L. Reid and Y-K. M. Peng, *"Potential Minimum Cost of Electricity of Superconducting Coil Tokamak Power Reactors"*, Proceedings of 13th IEEE Symposium on Fusion Engineering, Knoxville, Tennessee, October 1989, p. 258

[36] J. Sheffield et al., *"Cost Assessment of a Generic Magnetic Fusion Reactor"*, *Fusion Technology* **9** (1986) 199

[37] S. Thompson, *"Systems Code Cost Accounting"*, memo FEDC-M-88-SE,-004 (1988)

[38] J. D. Galambos, L. J. Perkins, S. W. Haney and J. Mandrekas, *Nuclear Fusion*, **35** (1995) 551

[39] J. P. Holdren et al., *"Report of the Senior Committee on Environmental Safety and Economic Aspects of Magnetic Fusion Energy"*, *Fusion Technology*, **13** (1988) 7

Appendix A

Non-optimisation Input File

The following is a typical input file used to run PROCESS in non-optimisation mode. Comments in [...] have been added to the right of each line.

```
* Numerics information                [Comment]

NEQNS = 14,                           [Number of active constraint equations]
NVAR = 14,                            [Number of active iteration variables]
ICC =   1,  2, 10, 11, 7, 16, 24, 5, 31, 32, 33, 34, 35, 36,  [Constraint eqns]
ixc =   5, 10, 12, 3,  7,  6, 36, 9, 48, 49, 50, 51, 53, 54,  [Iteration variables]
IOPTIMZ = -1,                         [Turn off optimisation]
ISWEEP=0                              [No scans (non-optimisation mode)]

* F-values and limits

FBETATRY = 1.0                        [N.B. active iteration variable 36]

* Physics parameters

ASPECT = 3.5,                         [Machine aspect ratio]
BETA = 0.042,                         [N.B. active iteration variable 5]
BT = 6.,                              [Toroidal field on axis]
DENE = 1.5e20,                        [N.B. active iteration variable 6]
FVSBRNNI = 1.0,                       [Non-inductive volt seconds fraction]
DNBETA = 3.5,                         [Troyon g coefficient]
HFACT = 2.,                           [N.B. active iteration variable 10]
ICURR = 4,                            [Use ITER current scaling]
ISC = 6,                              [Use ITER 89-P confinement time scaling law]
IINVQD = 1,                           [Use inverse quadrature]
IITER = 1,                            [Use ITER fusion power calculations]
ISHAPE = 0,                           [Use input values for KAPPA and TRIANG]
KAPPA = 2.218,                        [Plasma elongation]
Q = 3.0,                              [Edge safety factor]
RMAJOR = 7.0,                         [N.B. active iteration variable 3]
RNBEAM = 0.0002,                      [N.B. active iteration variable 7]
TBURN = 227.9                         [Burn time]
TE = 15.,                             [Electron temperature]
TRIANG = 0.6                          [Plasma triangularity]

* Current drive parameters

IRFCD = 1,                            [Use current drive]
IEFRF = 5                             [Use ITER neutral beam current drive]
FEFFCD = 3.,                          [Artificially enhance efficiency]
```

```
* Divertor parameters

ANGINC=0.262,                          [Angle of incidence of field lines on plate]
PRN1=0.285                             [Density ratio]

* Machine build

BORE = 0.12,                           [Machine bore]
OHCTH = 0.1,                           [OH coil thickness]
GAPOH = 0.08,                          [Inboard gap]
TFCTH = 0.9,                           [Inboard TF coil leg thickness]
DDWI = 0.07,                           [Vacuum vessel thickness]
SHLDITH = 0.69,                        [Inboard shield thickness]
BLNKITH = 0.115,                       [Inboard blanket thickness]
FWITH = 0.035,                         [Inboard first wall thickness]
SCRAPLI = 0.14,                        [Inboard scrape-off layer thickness]
SCRAPLO = 0.15,                        [Outboard scrape-off layer thickness]
FWOTH = 0.035,                         [Outboard first wall thickness]
BLNKOTH = 0.235,                       [Outboard blanket thickness]
SHLDOTH = 1.05,                        [Outboard shield thickness]
GAPOMIN = 0.21,                        [Outboard gap]
VGAPTF = 0,                            [Vertical gap]

* First wall, blanket, shield parameters

LBLNKT=0                               [Use old blanket model]
DENSTL=7800.                           [Steel density]

* TF coil parameters

OACDCP = 1.4e7,                        [N.B. active iteration variable 12]
ITFSUP = 1,                            [Use superconducting TF coils]
RIPMAX = 5.,                           [Maximum TF ripple]

* PF coil parameters

NGRP = 3,                              [Three groups of PF coils]
IPFLOC = 1,2,3,                        [Locations for each group]
NCLS = 2,2,2,1,                        [Number of coils in each group]
COHEOF = 1.85e7,                       [OH coil current at End Of Flat-top]
FCOHBOP = 0.9,                         [OH coil current at Begin. Of Pulse / COHEOF]
ROUTR = 1.5,                           [Radial position for group 3]
ZREF(3) = 2.5,                         [Z position for group 3]
OHHGHF = .71                           [Height ratio OH coil / TF coil]

* Vacuum system parameters

NTYPE = 1                              [Use cryopump]

* Heat transport parameters

ETATH=0.35                             [Thermal to electric conversion efficiency]
FMGDMW = 0.                            [Power to MGF units]
BASEEL=5.e6                            [Base plant electric load]
ISCENR=2                               [Energy store option]

* Buildings

FNDT = 2.                              [Foundation thickness]
EFLOOR=1.d5                            [Effective total floor space]
```

```
* Costs

IREACTOR = 1,                          [Calculate cost of electricity]
IFUELTYP = 0                           [Treat blanket, first wall etc as capital cost]
UCHRS = 87.9,                          [Unit cost of heat rejection system]
UCCPCL1 = 250,                         [Unit cost of high strength tapered copper]
UCCPCLB = 150                          [Unit cost of TF outer leg plate coils]
```

Appendix B

Optimisation Input File

The following is a typical input file used to run PROCESS in optimisation mode. Comments in [...] have been added to the right of each line.

```
* Numerics information                [Comment]

boundl(1) = 2.5,                     [Bound on iteration variable 1 (aspect)]
BOUNDU(10) = 3.,                     [Bound on iteration variable 10 (hfact)]
BOUNDU(60) = 4.d4,                   [Bound on iteration variable 60 (cpttf)]
NEQNS = 15,                          [Number of active constraint equations]
NVAR = 25                            [Number of active iteration variables]
ICC =   1,  2, 10, 11, 7, 16, 8, 24, 31, 32, 33, 34, 35, 36, 14, [Constraint eqns]
ixc =   5, 10, 12, 3,  7,        36, 48, 49, 50, 51, 53, 54, 19, [Corresponding...]
        1, 2, 4, 6, 13, 16, 29, 56, 57, 58, 59, 60,              [...iteration variables]
IOPTIMZ = 1,                         [Turn on optimisation]
MINMAX = 6,                          [Minimise cost of electricity]

ISWEEP=7,                            [Seven point scan]
NSWEEP=6,                            [Use WALALW as scanning variable]
SWEEP= 6.0,5.5,4.5,4.0,3.5,3.0,2.5   [Values of WALALW for each scan point]

* F-values and limits

FBETATRY = 1.0                       [N.B. active iteration variable 36]

* Physics parameters

ASPECT = 3.5,                        [N.B. active iteration variable 1]
BETA = 0.042,                        [N.B. active iteration variable 5]
BT = 6.,                             [N.B. active iteration variable 2]
DENE = 1.5e20,                       [N.B. active iteration variable 6]
FVSBRNNI = 1.0,                      [Non-inductive volt seconds fraction]
DNBETA = 3.5,                        [Troyon g coefficient]
HFACT = 2.,                          [N.B. active iteration variable 10]
ICURR = 4,                           [Use ITER current scaling]
ISC = 6,                             [Use ITER 89-P confinement time scaling law]
IINVQD = 1,                          [Use inverse quadrature]
IITER = 1,                           [Use ITER fusion power calculations]
ISHAPE = 0,                          [Use input values for KAPPA and TRIANG]
KAPPA = 2.218,                       [Plasma elongation]
Q = 3.0,                             [Edge safety factor]
RMAJOR = 7.0,                        [N.B. active iteration variable 3]
RNBEAM = 0.0002,                     [N.B. active iteration variable 7]
TBURN = 227.9                        [Burn time]
```

```
TE = 15.,                           [N.B. active iteration variable 4]
TRIANG = 0.6                        [Plasma triangularity]

* Current drive parameters

IRFCD = 1,                          [Use current drive]
IEFRF = 5                           [Use ITER neutral beam current drive]
FEFFCD = 3.,                        [Artificially enhance efficiency]

* Divertor parameters

ANGINC=0.262,                       [Angle of incidence of field lines on plate]
PRN1=0.285                          [Density ratio]

* Machine build

BORE = 0.12,                        [N.B. active iteration variable 29]
OHCTH = 0.1,                        [N.B. active iteration variable 16]
GAPOH = 0.08,                       [Inboard gap]
TFCTH = 0.9,                        [N.B. active iteration variable 13]
DDWI = 0.07,                        [Vacuum vessel thickness]
SHLDITH = 0.69,                     [Inboard shield thickness]
BLNKITH = 0.115,                    [Inboard blanket thickness]
FWITH = 0.035,                      [Inboard first wall thickness]
SCRAPLI = 0.14,                     [Inboard scrape-off layer thickness]
SCRAPLO = 0.15,                     [Outboard scrape-off layer thickness]
FWOTH = 0.035,                      [Outboard first wall thickness]
BLNKOTH = 0.235,                    [Outboard blanket thickness]
SHLDOTH = 1.05,                     [Outboard shield thickness]
GAPOMIN = 0.21,                     [Outboard gap]
VGAPTF = 0,                         [Vertical gap]

* First wall, blanket, shield parameters

LBLNKT=0                            [Use old blanket model]
DENSTL=7800.                        [Steel density]

* TF coil parameters

OACDCP = 1.4e7,                     [N.B. active iteration variable 12]
ITFSUP = 1,                         [Use superconducting TF coils]
RIPMAX = 5.,                        [Maximum TF ripple]

* PF coil parameters

NGRP = 3,                           [Three groups of PF coils]
IPFLOC = 1,2,3,                     [Locations for each group]
NCLS = 2,2,2,1,                     [Number of coils in each group]
COHEOF = 1.85e7,                    [OH coil current at End Of Flat-top]
FCOHBOP = 0.9,                      [OH coil current at Begin. Of Pulse / COHEOF]
ROUTR = 1.5,                        [Radial position for group 3]
ZREF(3) = 2.5,                      [Z position for group 3]
OHHGHF = .71                        [Height ratio OH coil / TF coil]

* Vacuum system parameters

NTYPE = 1                           [Use cryopump]

* Heat transport parameters

ETATH=0.35                          [Thermal to electric conversion efficiency]
FMGDMW = 0.                         [Power to MGF units]
```

```
BASEEL=5.e6                          [Base plant electric load]
ISCENR=2                             [Energy store option]

* Buildings

FNDT = 2.                            [Foundation thickness]
EFLOOR=1.d5                          [Effective total floor space]

* Costs

IREACTOR = 1,                        [Calculate cost of electricity]
IFUELTYP = 0                         [Treat blanket, first wall etc as capital cost]
UCHRS = 87.9,                        [Unit cost of heat rejection system]
UCCPCL1 = 250,                       [Unit cost of high strength tapered copper]
UCCPCLB = 150                        [Unit cost of TF outer leg plate coils]
```

Appendix C

Source Code Documentation

The development of the PROCESS code since its shipment from Oak Ridge National Laboratory in April 1992 has been fully documented in the Project Work File [3]. Presented here is a list of Project Work File Notes as of April 16, 2013 that address various issues related to the source code.

- Documentation of each individual source routine is an ongoing task. A Work File Note will be produced as each routine is processed, with the eventual aim of bringing all these together into a single document, i.e. a future edition of this manual.
- Summary of work performed since April 1992 : Note 0160
- Code status (routine SCCS version numbers) : Note 0165
- SCCS (Source Code Control System) implementation for PROCESS: Note 0003
- Present code standard : Note 0160
- Future code standard (to be adhered to) : Note 0167
- Directory structure and location of all relevant files : Note 0168
- Proposed future work : Note 0160
- Pulsed power plant coding : Note 0189
- Nuclide inventory and activation (FISPACT) module : Notes 0195 and 0231
- New cost algorithms : Note 0224
- Stellarator model : Note 0246
- D-^{3}He tokamak model : Note 0265

Reference [3] has recently been superseded by a new Project Work File, F/MI/PJK/PROCESS:-

- Tight Aspect Ratio Tokamak model: Note 001
- Reversed Field Pinch model: Note 009
- Test cases: Note 019